HAL
archives-ouvertes.fr

Relaxed (m,k)-firm Constraint to Improve Real-time Streams Admission Rate under Non Pre-emptive Fixed Priority Scheduling

Jian Li, Ye-Qiong Song

▶ To cite this version:

Jian Li, Ye-Qiong Song. Relaxed (m,k)-firm Constraint to Improve Real-time Streams Admission Rate under Non Pre-emptive Fixed Priority Scheduling. 11th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA'06,, Sep 2006, Prague, Czech Public, 2006. <inria-00108422>

HAL Id: inria-00108422
https://hal.inria.fr/inria-00108422

Submitted on 20 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Relaxed (m,k)-firm Constraint to Improve Real-time Streams Admission Rate under Non Pre-emptive Fixed Priority Scheduling

Li Jian
Jian.Li@loria.fr

Song Ye-Qiong
Song@loria.fr

LORIA-INPL - Campus Scientifique BP 239 - 54506 Vandoeuvre-lès-Nancy - France

Abstract

Comparing with hard real-time approach, (m,k)-firm constraint and its related scheduling policies are considered as an efficient way to increase the admission rate of real-time streams to a network thanks to the possibility to drop until k-m out of any k consecutive processing requirements, reducing thus the workload. Although it is interesting for probabilistic (m,k)-firm guarantee, we show however in this paper that for deterministic (m,k)-firm guarantee, reducing the workload by a factor of m/k does not contribute to reducing the resource requirement in general. So the relaxed (m,k)-firm constraint is proposed. The sufficient schedulability condition under non pre-emptive fixed priority scheduling is derived and its practical interest in terms of the resource requirement reduction is demonstrated.

1. Introduction

Priority queues are commonly implemented in network switches/routers and some of network interface cards (e.g. CAN controllers). It is so possible to use the fixed priority scheduling to adequately share a common resource (e.g. bandwidth) among a given set of packet streams for satisfying their different real-time constraints. In practice, for a given network bandwidth, the admission control mechanism must decide whether or not a set of input streams can be accept with the required real-time guarantee. For periodic or sporadic streams under deadline constraint on the transmission of each packet, classic worst-case response time calculation [1] [2] can be used for deciding the admissibility, although this method has trend to over sizing the needed resource because of the obligation to consider the worst-case.

In practice, many applications (e.g. multimedia transmission over Internet) may tolerant occasional deadline misses until a certain extent. (m,k)-firm constraint has been introduced [3] for further specifying this tolerance. A stream is said under (m,k)-firm constraint if the QoS degradation remains acceptable by the application as long as at least *m* out of any *k* consecutive packets can be transmitted before their deadline. In this paper, a packet which cannot be transmitted before its deadline is simply dropped, reducing thus the workload of the system. With such (m,k)-firm constraint instead of the hard real-time one, we can effectively increase the admission rate of the streams if only probabilistic (m,k)-firm guarantee is required [3], [4]. In fact, (m,k)-firm constraint can be considered as a kind of relaxation of the hard real-time (HRT) one, and HRT constraint can be considered as (k,k)-firm.

However, some applications may require deterministic rather than probabilistic (m,k)-firm guarantee. A typical example is the telesurgery where compressed video (using MPEG) and force feedback should be transmitted to the tele-surgeon in real-time. Although part of less important MPEG images could be dropped during network congestion [5], deterministic (m,k)-firm guarantee is still required for the safety regulation.

Deterministic (m,k)-firm guarantee has been addressed by many researchers [6, 7, 8, 9, 10, 11, 12]. Intuitively, reserving resources according to (m,k) requirement rather than (k,k) should reduce the necessary resource reservation. This is true for example for the case when flows are served by a WFQ server [5] or when only probabilistic (m,k)-firm guarantee is required. Unfortunately, it has been proven that in general, for achieving deterministic (m,k)-firm guarantee, one has to reserve, in general case, resources according to (k,k)-firm since the worst case must be considered. Moreover, the problem of non pre-emptive scheduling of N (m_i,k_i)-firm constrained flows ($i = 1, \ldots, N$) on a single resource (processor or network link) has been proved NP-hard in strong sense, such that no optimal scheduling can be expected under such model. In our previous work [12], we have given a way to better use the resource under (m,k)-firm constraint, but we also showed its limit because of the above problem. This problem seriously compromises the practical interest of using the (m,k)-firm model for network resource management when deterministic guarantee is required. In fact, as we will explain in this paper, reducing the workload by a factor of m/k does not contribute to reduce the resource requirement in general. In another word, when deterministic (m,k)-firm guarantee is required, one cannot

hope to admit more streams under (m,k)-firm constraint than streams under (k,k)-firm constraint. As for the hard real-time scheduling, scheduling streams under (m,k)-firm constraint and without pre-emption also obliges us to consider the worst-case; and this worst-case is in most case determined by the packet inter-distribution in time axis between streams rather than by the workload. So in the worst case it is possible that the resource requirement for the real-time system with some loss tolerance is not less than that under hard real-time requirement.

Faced to this problem, two research directions are possible. The first one is to specialize the stream set. For instance, in [10], by specializing the stream model such that the packets of all the streams must have the same transmission time and the same period, a high admission rate can be achieved. However this so particular stream model cannot be directly applied to a wider context in which each stream could have its different packet length and period. The second way is to extend the (m,k)-firm constraint. In [13], the deadlines of packets are relaxed to reduce the resource requirement, increasing thus a admission rate.

In this paper we follow the second research direction and evaluate the interest of relaxing the (m,k)-firm constraint.

Relaxed (m,k)-firm or R-(m,k)-firm constraint is first introduced in [14] [15] to resolve the above-mentioned problem. As with (m,k)-firm guarantee, R-(m,k)-firm provides the guarantee on the transmission delay of at least m out of any k consecutive packets ($m \leq k$). Instead of imposing a transmission delay constraint on each packet (i.e. deadline), R-(m,k)-firm only considers a global transmission delay constraint on a group of any k consecutive packets.

The rest of the paper is organized as follows. Section 2 presents the system model, useful definitions and show the low admission rate problem of scheduling streams under (m,k)-firm constraint which motivated us to do this work. Section 3 briefly describes the R-(m,k)-firm constraint, its possible applications and how it is situated comparing with other real-time relaxation schemes. Section 4 gives the sufficient schedulability condition under non pre-emptive fixed priority scheduling which can be used for admission control. Section 5 presents an numerical example showing the practical interest of the proposed R-(m,k)-firm constraint in terms of the admission rate increase. Finally we summarize our main contributions in section 6.

2. System model and problem definition

2.1. System model

Consider the following MIQSS (Multiple Input Queues Single Server) model (Fig. 1) where a set of n sreams: $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ share a single server. A stream generates jobs (e.g. packets) requiring a processing at the server (e.g. transmission time of a packet). The server should provide the service according to the streams' real-time constraints.

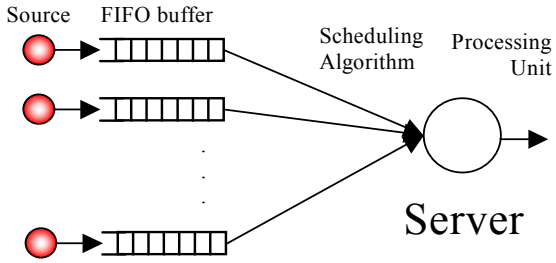

Figure 1. MIQSS model

2.2. Job model, workload and server utilization

Usually, real-time stream can be classified into periodic and sporadic. Periodic stream is of which all instance timing can be precisely known once after the stream release time. On the contrary, sporadic stream cannot be predicted, while the next instance can occur at any time after the finishing of previous one. In other words, the sporadic stream has a minimum inter-arrival time, and is invoked only upon the time of its last invocation. Typically, a stream τ_i can be modeled by $(c_i, p_i, d_i, m_i, k_i)$. A job (also called an instance in the following of this paper) of τ_i loads a quantity of work B_i (e.g. number of bits in a packet), which can be processed by the server within time c_i. Let R denote the processing capacity (e.g. bandwidth in bits/s), then following relation exists:

$$c_i = B_i / R. \qquad (1)$$

p_i is the period or the minimum inter-arrival time for periodic or sporadic stream. d_i is the deadline by which the instance should be completed by the server. Each stream τ_i is assumed to be under a (m_i, k_i)-*firm* constraint ($m_i < k_i$).

The *workload* of periodic stream set under HRT constraint and (m,k)-firm constraint are represented by formula (2) and (3), respectively:

$$\sum_{i=1}^{n} \frac{B_i}{p_i} \qquad (2)$$

$$\sum_{i=1}^{n} \frac{m_i}{k_i} \frac{B_i}{p_i} \qquad (3)$$

Note that sporadic stream's arrival rate is always below the periodic stream's, such that sporadic stream's workload is always smaller than the formula (2) and (3).

For comparing the efficiency of different scheduling policies, we also define *utilization* of the system as the ratio of the workload and the resource requirement (e.g. network bandwidth), such that the utilization under HRT

constraint and (m,k) constraint are denoted by formula (4) and (5), respectively:

$$\left(\sum_{i=1}^{n}\frac{B_i}{p_i}\right)/R=\sum_{i=1}^{n}\frac{c_i}{p_i} \qquad (4)$$

$$\left(\sum_{i=1}^{n}\frac{m_i}{k_i}\frac{B_i}{p_i}\right)/R=\sum_{i=1}^{n}\frac{m_i}{k_i}\frac{c_i}{p_i} \qquad (5)$$

For periodic or sporadic streams, the difficulty of scheduling non pre-emptive jobs can be affected by the occurrence time of first instance, or release time. Afterwards, once all streams have been given a no-negative value of release time, we say that a stream set is instantiated to a concrete stream set, and the concrete stream set is generated from the general stream set [15].

Through out this paper, time is assumed to be discrete and clock ticks are indexed by the natural number. Instances or invocations of stream begin and end at the clock ticks. Moreover, as ubiquitously implemented in networks and message transmission model, non-preemptive of instances are addressed in the analysis of scheduling, that is to say, once an instance occupies the server (e.g. a transmission begins), it should be finished without interruption.

2.3. Definition of (m,k)-firm and window-constraint.

The key idea we exploit here is to take the advantage of the "natural" packet loss tolerance of a large class of real-time applications to reduce the sufficient bandwidth reservation. If we consider applications such as video-on-demand, IP telephony, Internet radio, etc., many of them can tolerate packet losses to some extent.

Let us take the example of the voice packet transmission of the IP telephony service. Instead to guarantee the reliable transmission of all the packets with a large delay (as TCP), it is preferable to drop a voice packet if it cannot be transmitted in time (typically 400ms for IP telephony). (m,k)-firm model can then be used to specify such kind of tolerance by introducing the graceful QoS degradation between satisfying all packets' deadlines (i.e. (k,k)-firm guarantee) and (m,k)-firm guarantee.

A stream with (m,k)-firm deadlines is characterized by two parameters m and k such that at least m instances among any k consecutive ones must meet their deadlines.

Window Constraint is first proposed in [8], it requires the loss-tolerance of a stream such that x is the maximum acceptable loss in a fixed given window y. Dynamic Window Constraint Scheduling (DWCS) is also proposed to respect window-constraint. As mentioned in [8], window-constraint can be transformed to (m,k)-firm constraint, and vice verse. So in what follows, we will only focus on the (m,k)-firm constraint.

2.4. Problem definition

Predictability is necessary in real-time scheduling, which is often achieved by resource reservation and admission control under a priori assumed workload. It is always interesting to pre-dimension the least resource requirement for a given stream set. That is to say, for a given general stream set, it is desired to find the resource requirement such that all generated concrete scenarios will be schedulable.

Unfortunately, the results are not satisfying in terms of utilization until now, since deterministic guarantee can only be achieved by an enormous over-provisioned resource reservation.

In what follows, we will analyze the reasons which cause the low utilisation with some obvious examples. The showed problems lead us to propose a novel real-time constraint, which can relax the current hard real-time over-dimensioning problem.

Briefly, we summarize the non pre-emptive fixed priority scheduling difficulties in following three phrases. All phrases have their theoretic supports which are well studied in [15, 16, 17, 18]. In the paper, we will only demonstrate some phenomenon to show their omnipresence.

Phrase 1: *the* resource *requirement depends not only on the workload of the stream set, but also on the inter-distribution of streams' instances*.

Phenomena 1: for the same general stream set (the same period and execution time), the streams might be schedulable in one scenario but not be schedulable in another.

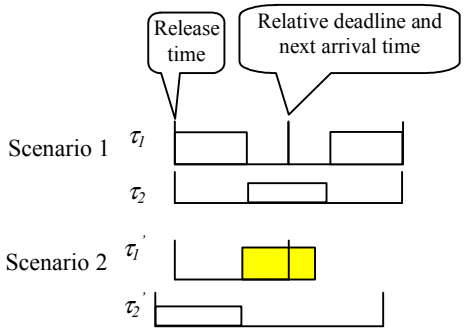

Figure 2. Phenomenon 1

For example, in Fig 2, if two streams are released at the same time, then they are schedulable as in concrete scenario 1. In contrast, if the release time of the τ_2' advances a little as in scenario 2, they are on longer schedulable. The process that advances the arrival time of the first instance of τ_2 by contrast of τ_1 is called as the change of the *inter-distribution of streams instances*. Clearly, in phenomenon 1, the same stream set has the

same workload, but its schedulability depends more on concrete scenario (different inter-distribution of streams' instances). In order to make scenario 2 to be schedulable, it needs more resource reducing thus the utilization.

More pessimistically, the dimensioning according to the concrete scenario is a problem of NP-hard in strong sense [15].

Phrase 2: *a stream set* with *less workload might be more* difficult *to schedule.*

Similarly, we will use an example to demonstrate the phrase 2.

Phenomena 2: for two stream sets, τ_2' in scenario 2 has doubled period and 1.1 time of execution time in contrast with τ_2 in scenario 1. Then, the workload of the second stream set is inferior to that of the first stream set. However, in Fig. 3, scenario 2, stream 1 is schedulable and stream 2 is not. This demonstrates that a stream set with less workload is not more susceptible to be schedulable. Theoretically [15, 16, 17, 18], this result is so pessimistic that a stream set with an arbitrary low workload could still be non-schedulable.

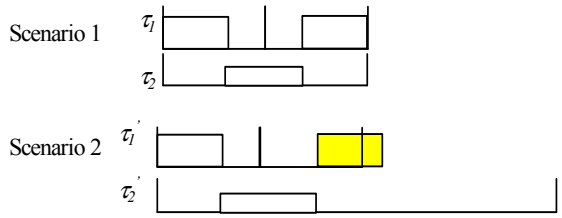

Figure 3. Phenomenon 2

As what has been explained above, current applications can tolerate some deadline misses, so that (m,k)-firm real time is proposed to resolve the over-provisioning problem. However, another phenomenon will show the pessimism of the (m,k)-firm constraint.

Phrase 3: *taking into account the worst-case, for a stream set under (m, k)-firm constraint, its resource requirement cannot* deterministically *be reduced in general case comparing with the same stream set under HRT.*

It can be demonstrated by Phenomenon 3: in Fig 4, a stream set of two streams, with (2,3)-firm constraint, is not schedulable in the scenario. In whatever scenario, the two streams require always to finish one instance of each stream in one period, such that if (2,3)-firm constraint is satisfied, (3,3)-firm constraint will also be satisfied. This fact obliges to reserve resource according to HRT constraint in order to guarantee an (m,k)-firm constraint. This is quite undesirable.

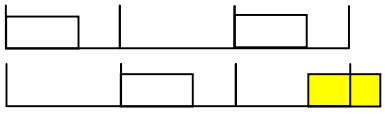

Figure 4. Phenomenon 3

Above all, it is sick to conclude that over-provisioning problem is inevitable in order to get deterministic guarantee. Moreover, with some loss tolerance such as (m,k)-firm constraint, the resource reservation can neither be improved for deterministic guarantee.

These facts motivated us to modify the current real-time constraint in order to ameliorate the utilization. In the sequel, we will propose a novel real-time constraint and analyze its gain under Fixed Priority scheduling.

3. Relaxed (m,k)-firm constraint

The practical advantage of *(m,k)-firm* constraint is to increase as much as possible the utilization factor of a stream set. As mentioned in the previous section, until now, there is not a non pre-emptive scheduling system which can get an interesting utilization for a general stream set under (m,k)-firm constraint. So that we will propose a relaxed (m,k)-firm real-time constraint to resolve this low utilization problem.

3.1. Definition of R-(m,k)-firm constraint

In any time interval [*s, t*] (with *t-s* $\geq l$, *where l is the minimum time length in which k unite workload can be sent*), a stream submitted *k* units of workload to a server. The server should finish the execution of at least *m* among them before time $t+\Delta$, where Δ is the maximum tolerable delay for the group of *k* units.

In fact, this new real-time constraint is a very general model [15], but in this paper, we will only analyze the case of periodic or sporadic stream. So that we will specify the definition with the example in Fig. 5 as follows:

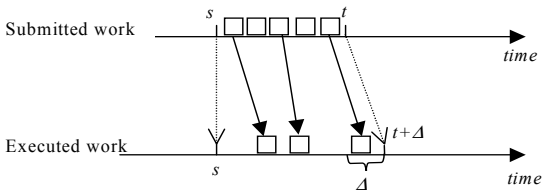

Fig. 5: R-(3,5)-firm constraint

1. In a certain time interval *[s,t]*, the stream can at most generate *k* instances. In other words, time point *t* is the end of generations of k_{th} packets. Obviously, for a periodic stream under R-(3,5)-firm, *[s,t]* = *5p*, such as in Fig 5.
2. The real-time constraint includes two factors: (m,k) factor and delay factor.
3. At least *m* among *k* instances should be executed within the delay constraint Δ in every fixed (no-overlap) window.
4. No-overlapping window just requires that after the release of a stream, for at least 3 instances are executed among the first 5 instances, as well as at

least 3 instances must be executed among 6_{th} to 10_{th} instances, and so forth.

We only showed a simple example scenario under R-(3,5)-firm and (3,5)-firm constraints, but it is not to say that all transmissions could tolerate the loss-rate of 40%. After all, *m* and *k* of *R-(m,k)*-firm constraint can be configured as any natural number. The configurations should be done according to the specified communication requirement.

3.2. Applications of R-(m,k)-firm

It is obvious that R-(m,k)-firm constraint is more flexible than the (m,k)-firm one. Although there are some deadline misses, it can be acceptable for a real-time multimedia communication such as VoIP, VoD, as well as some networked control systems where over-sampled data are transmitted by a network [14].

For example, consider a MPEG transmission stream, its transmission order differs from that for file storage or display. At the receiver side, some processing must be done after the reception of a group of picture (GOP) composed of I, P and B frames (i.e., decode) in order to display the video on screen or store to the disk. This is just the essence of R-(m,k)-firm constraint. In other words, per-packet deadline is not necessary for multimedia transmission; since the previous packet must wait for the following packets even it can arrive at time. On the other hand, all current multimedia transmissions have their individual attributes. For instance, MP3, the audio compression of MPEG, has not consecutive relation among the packets. Therefore, all packets (instances) will be treated equally, and the relationship among the packets in the application level could be ignored.

In addition, packets in a MPEG GOP are related. I frames are the most important ones comparing to P and B frame. B frames can be dropped according to a fixed (m,k)-pattern while still maintain an acceptable QoS degradation [5]. This differentiates the packets as mandatory and optional ones. Mandatory packets must be serviced (be sent in time) while optional packets could be dropped.

In this paper, the utilization will be shown using fixed pattern under Fixed Priority scheduling.

3.3. Other real-time constraint relaxation models

In fact, the deadline of instance is not the holy grail of real-time communication, and there have been a lot of work that tries to relax the deadlines to gain higher utilization, such as frame-based model [20], Pin-wheel scheduling [21] and virtual deadline window scheduling [13], etc.

The frame-based model defines a set of streams, which is to execute within each frame (time window) and is to complete before the end of frame. The problem is to schedule the stream set in a single frame with deadline D. Frame-based model assumes that all stream instances are stored in a single queue at the beginning of the frame (i.e. released at the same time at the beginning of the time window). In this case, the utilization of processor can reach 100%. Clearly, if the total streams' instances size is inferior or equal to the frame size, any non-idle scheduling can deal with them without any constraint violation. The scheduling in a frame can be repeatedly to do with infinite stream instances.

Note that frame-based model removes the deadlines of instances to provide a scheduling in the same frame size for all streams. However, for current diverse applications, streams demand different frame sizes. This makes the frame-based model not interesting for practical use in the QoS control in networks.

In our opinion, Pinwheel model is more interesting to reserve different quantity of time in different frame size for each stream. The generalized pinwheel scheduling problem is an offline scheduling for satellite-based communication as follows: Given a multiple set $\{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ of ordered pairs of positive integers, determine an infinite sequence over the symbols {1, 2, 3, …, n} such that, for each i, $1 \leq i \leq n$, any subsequence of b_i consecutive symbols contains at least a_i times of i_s. Differing from frame-based model, Pinwheel guarantees the execution time in a sliding frame.

Together, Pinwheel model and frame-based model can find optimal scheduling scheme, and their utilization factor can arrive to 100%. However, as mentioned, frame-based model constraint all streams in a unique frame, on the other hand, Pinwheel is designed for satellite-based communication model which only concerns unit size execution time. They cannot service current diverse multimedia applications.

Similar to R-(m,k)-firm constraint, Zhang and West [13] proposed a relaxed window constraint to gain utilisation. It allows stream instances to be serviced after their deadlines, as long as it can guarantee a minimum fraction of service to a stream in a fixed window. Its scheduling mechanism lengthens the instances' deadlines, and the deadlines are modified according to the execution time. Moreover, as like as DWCS, virtual deadline scheduling requires specially that the streams should have the unit size execution time and their period must be the multiple of the execution time. Notice that this model can be regarded as a special version of R-(m,k)-firm constraint in case that R-(m,k)-firm scheme requires (m,k) factor in no-overlapping window (fixed window) with delay factor Δ=0.

Briefly, R-(m,k)-firm synthesized all the above mentioned relaxation strategies to define a more general scheme for general task set.

4. Sufficient condition for *R-(m,k)-firm* system

To show that this novel real-time system can effectively economize the resource requirement, we will address a task set with fixed priority. In the task set, Γ =

$\{\tau_1, \tau_2, ..., \tau_n\}$, the priority is assigned according to the index, that is, the smaller the task index is, the higher priority it has.

4.1. Dimensioning method in traditional real-time deadline constraint

In [1] [2], the worst-case response time was studied for a given general task set under Non-preemptive fixed-priority scheduling. The task set can be judged as schedulable or not, according to whether the worst-case response time is inferior to the deadline or not.

$$r_i = \max_{q=0...Q}\{w_{i,q} + C_i - qP_i\} \quad \text{where}$$

$$w_{i,q} = qC_i + \sum_{j \in hp(i)} \left\lceil \frac{w_{i,q} + \gamma_{res}}{P_j} \right\rceil C_j + l_i$$

$$l_i = \max_{u \in lp(i)}\{C_u\}$$

The non-preemption is taken into account, l_i, which is the block factor from the task with lower priority. Q is the smallest value such that $w_{i,q} + C_i \le (Q+1)P_i$, γ_{res} is the resolution with which time is measured, and hp(i) is the subset of tasks with higher priority than task τ_i. P_i is the period length, and P_j is the period length of a tack $\tau_i \in hp(i)$.

However, in this paper, we are interested in finding the necessary resource for a given general task set, so that the execution time of every packet is an unknown argument. Nonetheless, every packet's maximum size is known, and according to formula (1), the execution time is transformed as the ratio of packet size and resource capacity. Hence, we will study the worst-case arrival workload in order to find the resource requirement. Such as follows:

For any $q \in (0...Q)$,

$$\left(w_{i,q} + B_i\right) / R \le \left(q+1\right) P_i \quad \text{where}$$

$$w_{i,q} = qB_i + \sum_{j \in hp(i)} \left\lceil \frac{w_{i,q} + \gamma_{res}}{P_j} \right\rceil B_j + l'_i$$

$$l'_i = \max_{u \in lp(i)}\{B_u\}$$

Clearly, the resource requirement can be measured by means of finding the smallest value of R which satisfies all:

$$(W_{i,q} + B_i) / (q+1)P_i = R \quad (q=0 \ldots Q)$$

Practically, this method calculates the sum of workload loaded by tasks with higher priority (e.g. in terms of bit) in the interval between the arrival time and the instance's deadline, as well as block factor form tasks with lower priority. A big enough resource capacity will reduce every execution time, such that all workload could be completed before deadline. Since the upper theorem calculates the workload in terms of the worst-case for a general task set, all concrete task set will be schedulable under HRT fixed priority scheduling.

In the same way, this method can be used to calculate the resource requirement for a task set under R-(m,k)-firm constraint.

In case of overload, we drop the instances according to the pattern proposed in [6], named as Sched_mkfirm, as below formula:

$$j = \left\lfloor \left\lceil \frac{j \times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor \qquad (1)$$

In this scheduling process, the j_{th} instance of task τ_i $i \in (1,n)$ will be loaded to the system if the formula (1) is satisfied, otherwise it will be dropped.

This Sched_mkfirm pattern ensures exactly that m_i out of any k_i instances will be loaded into the system. Moreover, from first instance of task τ_i, R-(m,k)-firm constraint requires that m_i instances must be finished by the end of every k_i periods.

4.2. Periodic Characters of Sched_mkfirm pattern:

Lemma1: counting from beginning of window, q_{th} (where $q \in (1, m_i)$) instance loaded to the system occurs at time $\left\lfloor (q-1)\frac{k_i}{m_i} \right\rfloor P_i$.

Lemma2: counting from the end of window, q_{th} (where $q \in (1, m_i)$) instance loaded to the system occurs at time $\left\lceil q\frac{k_i}{m_i} \right\rceil P_i$.

Lemma3: for a task τ_i, the most workload in a time interval of L is not bigger than:

$$\left\lceil \frac{L}{P_j}\frac{m_j}{k_j} \right\rceil C_j .$$

The proofs of Lemma1, Lemma2 and Lemma3 are trivial and not difficult. Moreover, they have been partly presented in [6], so that we will not detail the proofs in this paper.

4.3. Dimensioning for R-(m,k)-firm Constraint

Based on the dimensioning algorithms in HRT, we propose a sufficient condition, with which the system resource can be reserved to guarantee R-(m,k)-firm constraint.

Theorem: *sufficient condition*
For any general task set, if the following formulas are satisfied, any concrete task set generated from it will be schedulable under fixed priority Non-preemptive scheduling.

$$w_{i,q}(q=1...m_i) \le \left\lceil q\frac{k_i}{m_i} \right\rceil P_i * R$$

$$w_{i,q} = qB_i + \sum_{j\in hp(i)} \left\lceil \frac{\left\lceil q\frac{k_i}{m_i} \right\rceil P_i}{P_j} \frac{m_j}{k_j} \right\rceil B_j + l_i$$

$$l_i = \max_{u\in lp(i)}\{B_u\}$$

Proof: we prove this theorem by contradiction. Assume that time t_d is the first time when the system falls into failure state. The failure is caused by the task τ_i which violates R-(m_i,k_i)-firm constraint at time t_d. According to Lemma2, the last loaded instance of τ_i occurs at time $t_s = t_d - \left\lceil \frac{k_i}{m_i} \right\rceil P_i$, in which we should calculate the worst interference. To do this in $[t_s, t_d]$, three cases could be found:

Case 1: the previous loaded instance of task τ_i is completed before time point of t_s. l_i stands for the block factor due to Non-preemption, that is, a lower priority task's instance has occupied the server before t_s, so that it has the highest priority and must be finished at first. The worst interference caused by the tasks with higher priority (hp(i)) could be calculated according to Lemma 3, such that the sum of total workload from higher priority task can be denoted by:

$$\sum_{j\in hp(i)} \left\lceil \frac{\left\lceil \frac{k_i}{m_i} \right\rceil P_i}{P_j} \frac{m_j}{k_j} \right\rceil B_j$$

then, the most workload in $[t_s, t_d]$ can be described as:

$$w_{i,1} = C_i + \sum_{j\in hp(i)} \left\lceil \frac{\left\lceil \frac{k_i}{m_i} \right\rceil P_i}{P_j} \frac{m_j}{k_j} \right\rceil C_j + l_i$$

Because task τ_i violates R-(m_i,k_i)-firm constraint, in this case, $w_{i,1}$ must be bigger than the length of $[t_s, t_d]$, while it contradicts with the theorem when $q=1$.

Case 2: the previous loaded instance of task τ_i is completed just at time t_s. In this case, there is no longer the block factor caused by lower priority tasks. Moreover, the worst interference caused by higher priority task is the same as in the first case. Clearly, the sum of total workload in time interval $[t_s, t_d]$ is inferior than in the case 1. So, if the case 1 can be satisfied, case 2 will be well satisfied as well.

Case 3: the previous instance of task τ_i is completed after time t_s. In this case, we derive the workload in the time interval, when the last two instances are loaded into the system, such as $t_s = t_d - \left\lceil 2\frac{k_i}{m_i} \right\rceil P_i$.

For the calculation, the same three cases will be found as the above analysis. In addition, the last two instances of τ_i suffer only once a block factor in time interval $[t_s, t_d]$. If task τ_i is not suspended by itself, the sum of workload is described as:

$$w_{i,2} = 2C_i + \sum_{j\in hp(i)} \left\lceil \frac{\left\lceil 2\frac{k_i}{m_i} \right\rceil m_j}{P_j} \frac{}{k_j} \right\rceil C_j + B_i$$

This iteration analysis can stop until $\boldsymbol{q = m_i}$; in the other word, until the last m_{ith} loaded instance. Because the last $(m_i+1)_{th}$ loaded instance must have been finished before $t_d - \left\lceil m_i \frac{k_i}{m_i} \right\rceil P_i = t_d - k_i p_i$.

Otherwise, if it is finished after $t_d - k_i p_i$, the failure has already occurred in the previous window. This contradicts with the assuming that t_d is the first time of failure state.

End of proof.

5. Simulations

In this section, the sufficient condition will be used to provision a task set under R-(m,k)-firm constraint in comparison with (m,k)-firm constraint and HRT constraint. With the results, HRT over-provisioning problem will be shown, the advantages of R-(m,k)-firm will be shown in terms of resource requirement.

Assume one task set with three tasks, $\Gamma_1 = \{\tau_1, \tau_2, \tau_3\}$, and their parameters are shown in Table 1:

Γ_1	B_i (kb)	p_i (ms)	(m, k)
τ_1	1	2	(4, 5)
τ_2	4	5	(7, 8)
τ_3	5	8	(7, 9)

TABLE I. PARAMETERS OF TASK SET 1

We will calculate the workload to be executed in order to guarantee HRT, (m,k)-firm constraint and R-(m,k)-firm constraint, respectively.

In Fig. 5, the sums of workload at every time points are shown, before which the workload should be executed. The resource requirement can be represented by the slope of the beeline which just exactly bounds all the workload.

According to sufficient condition theorem, the most ratio of workload and its time length to be executed can be used to provision the system resource under FP scheduling for HRT and R-(m,k)-firm constraint, such as shown in Table 2.

Γ_1	Workload (Mps)	Resource (Mps)
HRT	1.95	3
(m,k)-firm	1.59	3
R-(m,k)-firm	1.59	1.75

TABLE II. PROVISIONING RESULTS FOR REAL-TIME CONSTAINTS

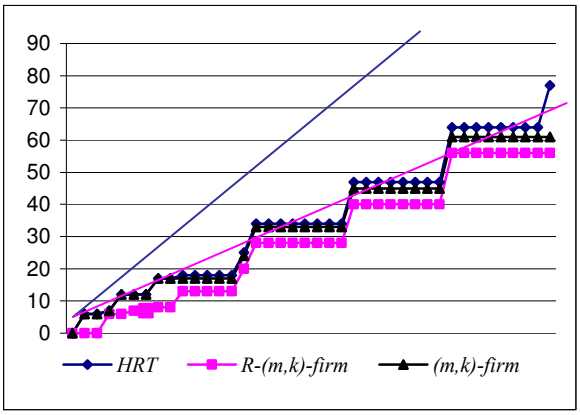

Figure 5. the arrival workload to be executed

In table 2, it is obvious that task set Γ_1 has a workload of 1.95Mps, and in order to guarantee its HRT constraint, 3Mps bandwidth is required. (m,k)-firm constraint can tolerate some degradation, such that its workload reduces to 1.59Mps. Although (m,k)-firm constraint has a less workload, its resource requirement is the same as HRT. This fact proves the phrase 3 in section 2.4. Rather than (m,k)-firm, Γ_1 under R-(m,k)-firm has the same workload as (m,k)-firm constraint, and it requires much less resource as well. Obviously, R-(m,k)-firm ameliorates this problem.

In addition, the over-provisioning problem is ameliorated as well be R-(m,k)-firm constraint by contrast with HRT and (m,k)-firm.

Deeper, the most resource requirement is occurred in the case where an instance τ_1 is blocked by an instance of τ_3 because of non-preemption.

Assume now that another task $\tau_4 = \{c_4, p_4, m_4, k_4\} = \{2, 5, 4, 6\}$ arrives to the system, and the available system resource currently is just 3Mps. Intuitively, τ_4 can not be admitted, but if we recalculate the resource requirement according to the sufficient condition, the provisioning is shown in Table 3.

Γ_2	Workload(Mps)	Resource (Mps)
HRT	2.325	3
(m,k)-firm	1.85	3
R-(m,k)-firm	1.85	2.4

TABLE III. PROVISIONING OF RESOURCE REQUIREMENT AND GAIN

Observe that with new task τ_4, a new task set Γ_2, requires the same resource as Γ_1 for HRT and (m,k)-firm constraint. Theoretically, the most resource is still required in the case where an instance τ_1 is blocked by an instance of τ_3. This fact demonstrates the problems proposed in phrase 1 and phrase 2 in section 2.4. However, our R-(m,k)-firm constraint requires still less resource than HRT and (m,k)-firm.

Actually, if task τ_2 finishes, the new task set Γ_3, will still requires the same resource for HRT and (m,k)-firm constraints, but Γ_3 with R-(m,k)-firm constraint will requires less resource. Hence, we can demonstrate that R-(m,k)-firm constraint requires the resource is monotonic with the workload, which is an expected result, and ameliorates the problems in Phrase 1 and 2 in section 2.4.

Until now, we have shown the advantage of R-(m,k)-firm in comparison of HRT and (m,k)-firm, and showed the achievement of motivations.

In order to show the schedulablity with above provisioning method, we will show the scheduling trajectory as shown in Fig 6, where clock tick is 100μs.

	c_i (ms)	p_i (ms)	(m, k)
τ_1	0.4	2	(4, 5)
τ_2	1.7	5	(7, 8)
τ_3	2.1	8	(7, 9)
τ_4	0.8	5	(4, 6)

TABLE IV. TRANFORME THE INSTANCE WORKLOAD TO EXECUTION TIME

Fig. 6 shows a scheduling process in 300ms which is departed into three lines because of space limit. In each part, from up to down, traces are the scheduling process of τ_1, τ_2, τ_3 and τ_4. Obviously, τ_1, τ_2 and τ_3 are executed without miss of traditional deadline, while in τ_4 scheduling process, there are miss of traditional deadline in every window, such that any deadline oriented scheme cannot be satisfied in this case. However, R-(4,6)-firm constraint of τ_4 is well guaranteed. This fact shows the graceful degradation provided by R-(m,k)-firm constraint.

6. Conclusion

R-(m,k)-firm constraint is first proposed in [15], which aimed to give a new real-time scheme to provide graceful degradation. R-(m,k)-firm is a global and flexible scheme, while in this paper, we focus on its advantage for schedule periodic task set.

An offline sufficient condition was given, which can find the least resource dimension in order to absolutely guarantee R-(m,k)-firm constraint. This dimensioning can considerably economize the resource in comparison of (m,k)-firm constraint. Moreover, the simulations demonstrate that with the allocated dimensioning resource, the task set R-(m,k)-firm constraints can be successfully and deterministically guaranteed. Also, the relaxation of R-(m,k)-firm by comparison of (m,k)-firm is also demonstrated by the simulation.

R-(m,k)-firm constraint is a novel real-time constraint which is suitable for various multimedia transmissions. Together with the sufficient resource allocation condition, R-(m,k)-firm can serve QoS guarantee for real-time transmissions in the network.

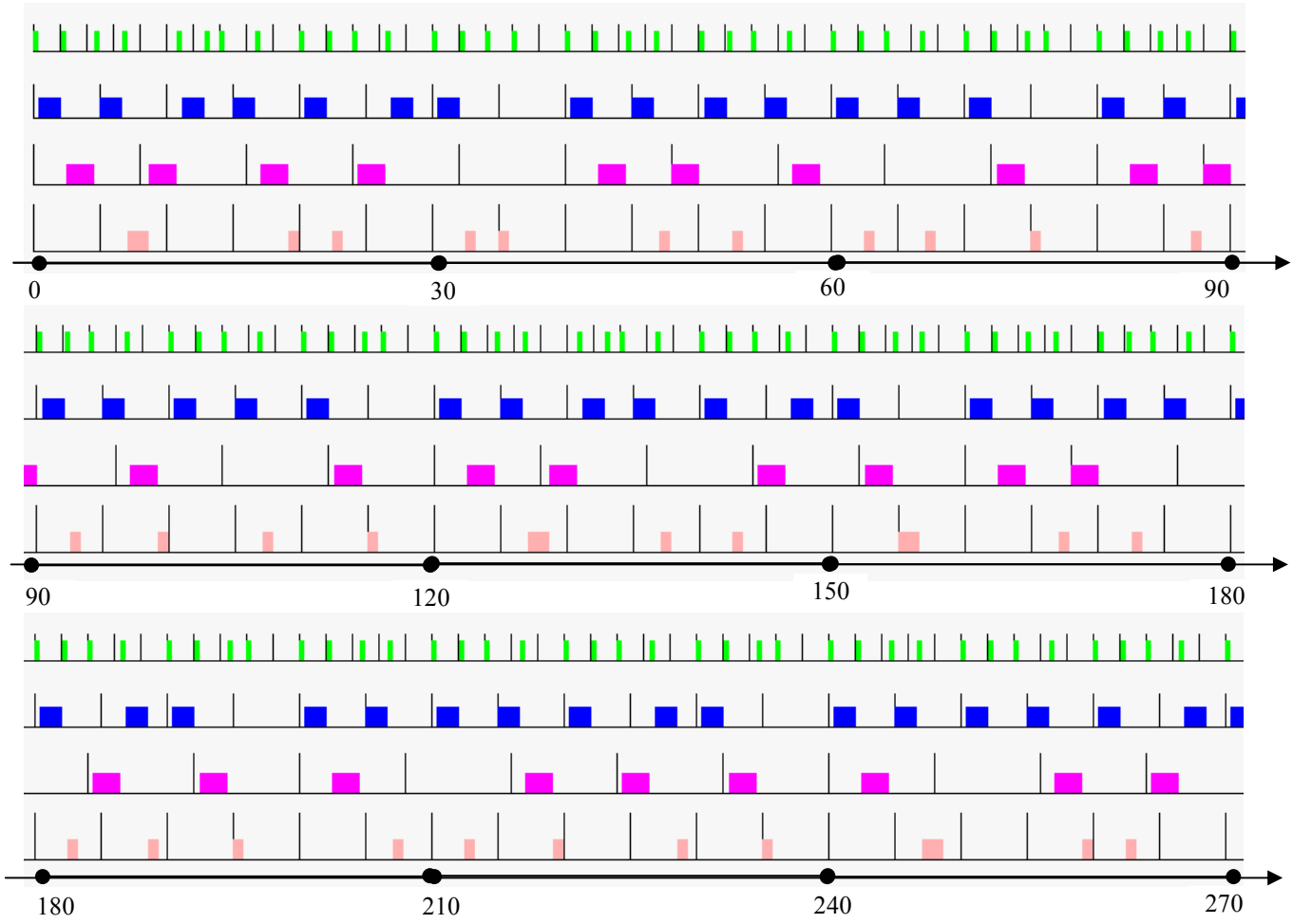

Figure 6. Scheduling trajectory under non-preemptive fixed priority for R-(m,k)-firm

References

[1] K. Tindell Fixed-Priority Scheduling of Hard Real-Time Systems. PhD thesis, University of York, UK, 1994.

[2] K. Tindell, A. Burns and A.J. Wellings "Analysis of Hard Real-Time Communications" Real-Time Systems, Vol. 9(2), pp. 147-171, Kluwer Academic Publishers (September 1995).

[3] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines", IEEE Transactions on Computers, 44(4), 1443–1451, Dec.1995.

[4] Moncef Hamdaoui, Parameswaran Ramanathan, "Evaluating Dynamic Failure Probability for Streams with (m, k)-Firm Deadlines," *IEEE Transactions on Computers*, vol. 46, no. 12, pp. 1325-1337, Dec., 1997.

[5] A. Koubâa, Y.Q. Song, "Graceful Degradation of Loss-Tolerant QoS using (m,k)-Firm Constraints in Guaranteed Rate Networks", *Computer Communications*, Vol.28, pp1393-1409, 2005.

[6] P. Ramanathan, "Overload management in real-time control applications using (m; k)-firm guarantee," IEEE Transactions on Parallel and Distributed Systems (Special issue on Dependable Real-time Systems), pp. 549-559, June 1999.

[7] G. Quan and X. Hu, "Enhanced Fixed-priority Scheduling with (m, k)-firm Guarantee", *Proc. Of 21st IEEE Real-Time Systems Symposium*, pp.79-88, Orlando, Florida, (USA), November 27-30, 2000.

[8] Richard West and Karsten Schwan, "Dynamic Window-Constrained Scheduling for Multimedia Applications", in Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS), 1999 .

[9] Mok, A.K. and W. R. Wang, "Window-Constrained Real-Time Periodic Task Scheduling", 22nd IEEE Real-Time Systems Symposium (RTSS'01), pp15-24, London, England, December 03 - 06, 2001.

[10] Richard West, Yuting Zhang, Karsten Schwan and Christian Poellabauer, "Dynamic Window-Constrained Scheduling of Real-Time Streams in Media Servers", IEEE Transactions on Computers, Volume 53, Number 6, pp. 744-759, June 2004

[11] Li, Jian and Song, YeQiong and Simonot-Lion, Françoise "Schedulability analysis for system under (m,k)-firm constraints." In 5th IEEE International Workshop on Factory Communication System - WFCS'2004, Vienne, Autriche, Sep 2004.

[12] Li, Jian and Song, YeQiong and Simonot-Lion, Françoise, "Providing real-time applications with graceful degradation of QoS and fault tolerance according to (m,k)-firm model" IEEE Transactions on Industrial Informatics, manuscript ID TII-05-12-0074.R2

[13] Yuting Zhang, Richard West and Xin Qi, "A Virtual Deadline Scheduler for Window-Constrained Service Guarantees", in

Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS), December 2004.

[14] Jian Li, YeQiong Song, "DLB: A Novel Real-time QoS Control Mechanism for Multimedia Transmission," *aina*, pp. 185-190, 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), 2006.

[15] Jian Li, YeQiong Song, "R-(m,k) firm: A novel QoS scheme for real-time flow guarantee in Networks" 14TH INTERNATIONAL CONFERENCE ON REAL-TIME AND NETWORK SYSTEMS, RTNS'06, Poitiers, France May 30-31, 2006

[16] K. Jeffay, D.F. Stanat, and C.U. Martel, "On Non-Preemptive Scheduling of Periodic and Sporadic Task", IEEE real-time systems symposium, pp129-139, San Antonio (USA), Dec. 4-6, 1991.

[17] R. West and C. Poellabauer, "Analysis of a Window-Constrained Scheduler for Real-Time and Best-Effort Packet Streams", Proc. of 21st IEEE Real-Time Systems Symposium, Orlando, Florida, (USA), November 27-30, 2000.

[18] J. LI. Sufficient Condition for Guaranteeing (m,k)-firm Real-Time Requirement Under NP-DBP-EDF Scheduling. Technical report No. A03-R-452, Stage de DEA, LORIA, Jun, 2003.

[19] L. Georges, P. Mühlethaler, N. Rivierre. "A Results on Non-Preemptive Real time Scheduling", INRIA research report N° 3926, may 2000.

[20] Frank Liberato, Sylvain Lauzac, Rami Melhem, Daniel Mosse. "Fault Tolerant Real-Time Global Scheduling on Multiprocessors", p.0252, 11th Euromicro Conference on Real-Time Systems, 1999.

[21] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. "The pinwheel: A real-time scheduling problem". In Proc. of the 22nd Hawaii International Conference on System Science, pages 693− 702, January 1989