A Graceful QoS Degradation Scheme for Loss Tolerant Real-time Applications

Jian Li, Ye-Qiong Song

A Graceful QoS Degradation Scheme for Loss Tolerant Real-time Applications

Li Jian, Song YeQiong

Campus Scientifique - BP 239
54506 Vandoeuvre-lès-Nancy Cedex, France
E-mail: {Jian.Li, Song}@loria.fr;

Abstract. This paper presents a new real-time Quality of Service (QoS) guarantee scheme called Relaxed (m,k)-firm denoted by R-(m,k)-firm. The problem we deal with is that for a set of streams (e.g. sources which generate packets) sharing a common resource, deterministic (m,k)-firm guarantee of each stream can lead to an arbitrarily low resource utilization, which makes (m,k)-firm useless for QoS guarantee in a network. The goal of the proposed constraint relaxation is to achieve a higher resource utilization while still deterministically guarantee the (m,k)-firm constraint. As with (m,k)-firm guarantee, it 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 per packet (i.e. deadline), R-(m,k)-firm only considers a global transmission delay constraint on a group of any k consecutive packets. This constraint relaxation may be acceptable for a large class of soft real-time applications such as multimedia flow transmissions in the networks for which occasional packet drops can be tolerated.

One of the possible implementations of the R-(m,k)-firm scheme is also provided with the development of a new traffic control mechanism, called Double-Leaks Bucket (DLB). DLB selectively drops a proportion of packets of a flow or an aggregated-flows in case of the network congestion while still guaranteeing the R-(m,k)-firm constraint. The sufficient condition for this guarantee is given for configuring the DLB parameters. Simulation results show the advantage in terms of avoidance of consecutive dropping and queue length.

Additionally, a comprehensive discussion on the existing constraint relaxation strategies is developed showing the generality of the R-(m,k)-firm scheme.

1. Introduction and Motivation

Internet nowadays supports more and more real-time and business-critical applications. However, for providing them with transmission delay guarantee, neither Intserv nor Diffserv consists in an efficient solution [1]. In fact, in order to provide a bounded transmission delay to a flow (Intserv) or a class of flows (Diffserv), because a flow often generates bursty traffic (case of most VBR applications and aggregated Diffserv class of flows), a bandwidth reservation policy according to the peak rate of the flow is an over-provisioning one. This leads to low resource utilization in average case. Another problem we should deal with in providing real-time QoS is the network congestion because of the router overload. This can occur when a path includes one or several routers which do not support Diffserv. As it may lead to packet drop, although occasional packet drops could be tolerated, long consecutive packet drops must be

avoided since it can drastically decease the QoS for applications such as audio/video diffusion. Unfortunately, existing queue management schemes such as TD (Tail-Drop) do not address the consecutive packet drop problem. RED (Random Early Detection) [2] has been proposed to deal with the problem with random dropping. Therefore, it does not give any guarantee on non-consecutive packet drops.

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. In fact, 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 packetized voice transmission of the IP telephony service. Instead to guarantee the reliable transmission of all the packets with a large delay, it is preferable to drop a voice packet if it cannot be transmitted in time (typically 400ms for IP telephony). The (m,k)-firm model is first introduced by Hamdaoui and Ramanathan [3], which can 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. QoS management according to the (m,k)-firm model has several advantages: (1) during network congestion, packets are dropped according to the (m,k) model rather than non-deterministically as the case of TD and RED. Thus undesirable long consecutive packet drops can be avoided; (2) the fact to aim at only (m,k)-firm guarantee instead of (k,k)-firm may require less resource reservation since the average workload is reduced by a factor of m/k for providing the minimum QoS level. This last point is interesting when congestion occurs or when the peak rate reservation (i.e., over-provisioning) cannot be provided at admission control.

At first glance, the (m,k)-firm model [3] seems to be an interesting one for resolving both of the problems of the resource over-provisioning and long consecutive packet drops. The (m,k)-firm model says that the deadlines of at least *m* out of any consecutive *k* packets must be met. Moreover packets whose deadline cannot be met are not transmitted (i.e. dropped), this is why we use the term "firm real-time" instead of "soft real-time". Notice that the term "any consecutive *k* packets" implies a sliding window guarantee for a flow. Some work exists in applying (m,k)-firm to QoS management. In [4], it proposed to use (m,k)-firm model instead of RED for congestion control and experimentally showed its interest. However nothing is provided concerning the transmission delay guarantee. [5] proposed an integration of the existing (m,k)-firm scheduling algorithms into the Diffserv architecture for providing average performance improvement.

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 [6] or when only statistic (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 resources according to (k,k)-firm since the worst case must be considered [7, 8, 9]. Moreover, the problem of non preemptive 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. This problem seriously compromises the practical interest of using the (m,k)-firm model for network resource management.

Faced to this low utilization problem (due to NP-hard), three research directions are possible. The first one is to look for the sub-optimal scheduling using heuristic methods. This is generally not suitable for on-line QoS control because of the long computing time. The second 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, the utilization factor is improved. However this so particular stream model cannot be directly applied to the multimedia transmission in which each stream could have its different packet length and period. The third way is to extend the (m,k)-firm one. In [11], the deadlines of packets are relaxed to reduce the resource requirement.

In this paper we follow the third research direction and propose to modify the concept of (m,k)-firm. Instead of considering the traditional guarantee of the individual packet deadline, we define a global deadline for any group of k consecutive packets. Formally, in an interval $[s, t]$, the source has sent k packets to the network, then the destination should be assured to receive at least m among them (delivered *in order or not*) before time $t+\Delta$, where Δ is the maximum tolerable transmission delay caused by the network for any group of k consecutive packets.

With this novel definition, it is obvious that the QoS requirement is given from per flow or per aggregated-flow point of view instead of the per packet deadline (m,k)-firm constraint. So we call this Relaxed (m,k)-firm guarantee and shortened by R-(m,k)-firm.

For implementing R-(m,k)-firm scheme, we also designed a new mechanism, called Double Leaks Bucket (DLB) for dropping a proportion of packets of a flow or a class of flows in case of network congestion while still guaranteeing the R-(m,k)-firm constraint of *(r, b)-bounded* [12] flows. Where r stands for the average arrival rate while b the burst. In [15] it has been shown that (r, b)-bounded can include periodic and sporadic flows (with or without jitters).

The R-(m,k)-firm scheme can be considered as one of the real-time constraint relaxation strategies similar to the Pinwheel model [13], frame-based model [14] and Virtual window constrained model [11]. We will show that R-(m,k)-firm is a general model which can include the previous ones.

The rest of this paper is organized as following. Section 2 describes in more detail the R-(m,k)-firm scheme. The DLB mechanism is presented in section 3. In section 4 we discuss on the existing constraint relaxation strategies and show the generality of the R-(m,k)-firm scheme. The performance comparison among DLB, TD and RED is presented in section 5. Then, the wide perspective adaptation of DLB is discussed in section 6. Section 7 summarizes our contributions.

2. R-(m,k)-firm scheme

2.1 Motivation of R-(m,k)-firm

Let us consider a traditional periodic or sporadic task[1] set described as $\Gamma=(\tau_1,...\ \tau_n)$. A task is described by $\tau_i = (c_i, p_i, d_i, m_i, k_i)$, where c_i stands for the execution time of an instance, p_i stands for the period or minimum inter-arrival time of instances, d_i is the relative deadline before which the instance must be completed, otherwise the instance will miss its deadline and in firm real-time, it is dropped directly without execution; and m_i, k_i describe that the deadline of at least m_i out of any consecutive k_i instances must be met.

The practical advantage of (m,k)-firm constraint is to increase at much as possible the utilization factor of a task set which is given in terms of $\sum_{i=1}^{n} \frac{m_i}{k_i}\frac{c_i}{p_i}$. Obviously, this utilization cannot be higher than 100%, such that the **gain** of a system is to improve the utilisation factor to 100% at most.

As mentioned in the previous section, until now, there is not a non pre-emptive scheduling system which can get an interesting gain for a general task 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.

2.2 Definition of R-(m,k)-firm QoS constraint

An R-(3,5)-firm constraint is shown in Fig. 1.

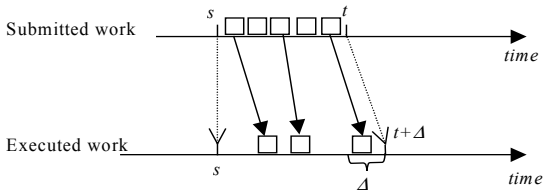


Fig. 1: R-(3,5)-firm constraint

Definition of R-(m,k)-firm constraint:
In any time interval $[s, t]$ (with $t\text{-}s \geq l$), a task submits k units of workload to a server. The server should finish the execution of at least m among them (*in order or not)* before time $t+\Delta$, where Δ is the maximum tolerable delay caused by the server for the group of k units.

[1] Here we use the term task for keeping close to the real-time scheduling terminology. However, it should understand that a task is a general term which can stand for a source generating packets in the context of networks

The following points help to understand this definition.

1) Any time interval means a sliding window, which can start from any time point, denoted by *s*.
2) A task τ_i can either be modelled by a periodic or sporadic source *(c_i, p_i, m_i, k_i)* or by a (r_i,b_i)-bounded source under *(m_i,k_i)-firm* constraint but with Δ as the deadline of any group of k consecutive instances starting at time *s*.
3) The constraint is given for each task: every task can require its own constraint without considering other tasks. The system should guarantee R-(m,k)-firm constraint individually for each task.
4) The real-time constraint includes two factors: (m,k) factor and delay factor.
5) (m,k) factor of constraint: at least *m* among *k* instances should be executed within the delay constraint Δ. In general case, (m,k) factor applies to the sliding window. However it can also applies to the non-overlapping fixed windows.

Delay factor Δ: this factor assures that (m,k) factor must be realised before a maximum delay after the end of the release of the k^{th} instance starting from time *s*. For the sliding window requirement, it requires that from no matter when one task generates k instances in time length not smaller than *l*, such that the system assures the execution of at least *m* instances before *l*+*Δ*. No-overlapping window just requires that after the release of a task, at least *m* instances are executed among the first *k* instances, as well as the *m* among $(k+1)^{th}$ to $2k^{th}$ instances, and so on.

2.3 Application in network

After the comprehension of R-(m,k)-firm constraint, we give an example in networks, as shown in Fig. 2. The source has a virtual stack, and it sends the packets from its stack head through the network. The destination has also a corresponding stack, and adds the received packets to the stack tail. Supposing in the time interval [*s, t*], *k* packets have been sent to the destination, the QoS provided by the network must assure the destination to receive at least *m* packets among the *k* for adding into its stack before *t*+*Δ*. There are *k*-*m* discardable packets in *k* consecutive ones.

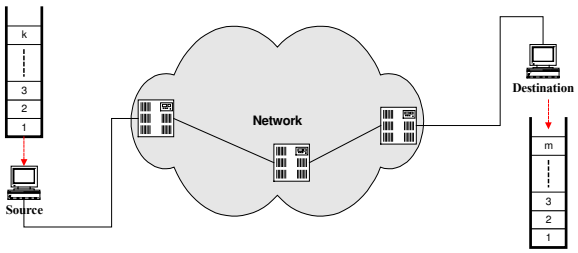


Fig. 2: Per Flow QoS

Actually, R-(m,k)-firm constraint replaces the conventional real-time constraint on the individual packet deadline.

2.4 Demonstration of R-(m,k)-firm advantages by Virtual scheduling pattern

Fig.3 shows the advantage of our R-(m,k)-firm constraint in contrast with conventional per packet deadline constraint. Each block stands for a packet.

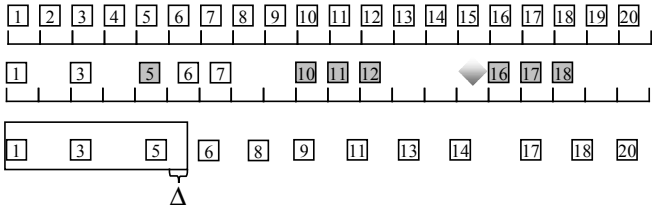


Fig. 3: Example of a virtual scheduling pattern under (3,5)-firm and R-(3,5)-firm constraint

The fist scenario (first line) shows a periodic packet stream that sends a packet at each beginning of period of P seconds. The second scenario (2nd line) shows a scheduling under (3,5)-firm constraint. In the second scenario, the server is obliged to serve all grey blocks. Otherwise, the system falls into failure state, such as the 15^{th} block. Once the system falls into failure state (i.e. the (m,k)-firm constraint is violated), the server is still obliged to serve the next 3 packets (16^{th}, 17^{th}, 18^{th}) to restore system. Note that this obliged service of packets, scheduled with other streams, will cause high resource requirement (or reduced utilization) forming the so-called interference point [7], [16]. The well distributed interference points could reduce the resource requirement [16]. However, the optimal distribution of those interference points is an NP-hard problem in strong sense, making it impossible to always reduce the over-provisioning problem. The third scenario shows a sequence under R-(3,5)-firm constraint, whose window size is configured according to R-(3,5) constraint with $5P+\Delta$. Readers can verify by sliding the window or by positioning the no-overlapping windows, and will find there are always at least 3 packets in the window no matter which beginning of period it is slide to. Although there are a lot of deadline misses, the sequence can still be accepted by R-(3,5)-firm.

2.5 R-(m,k)-firm is flexible and adaptive

It is obvious that our R-(m,k)-firm pattern 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.

In previous section, 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.

The R-(m,k)-firm scheme being specified, we propose in the next section a traffic control mechanism for deterministically guaranteeing R-(m,k)-firm constraint with high utilization factor. In fact satisfying R-(m,k)-firm constraint is not a trivial problem since the (m,k) factor and the delay factor are two antagonist factors for a given server. On the one hand, serving more instances (or packets) favours the (m,k) factor but leads to more delay. On the other hand, dropping more instances (or packets) may reduce the delay factor but risk to jeopardize the (m,k) factor.

3. DLB (Double-Leaks Bucket)

According to R-(m,k)-firm constraint, we develop one novel mechanism termed as DLB from the traditional leaky bucket [12]. DLB has two leaks named Serving Leak (SL) and Discarding Leak (DL) as depicted in Fig. 4.

3.1 Liquid model of DLB mechanism

Firstly, to simplify the problem, we start the analysis with a liquid model, whose workload is in terms of 'water' that can be split infinitesimally. The network should guarantee the 'water' that travels through SL, whilst the DL controlled by one switch gives the capacity to throw out the water from the bucket. With the service guarantee for SL, R-(m,k)-firm constraint could be satisfied. The water going through the DL is discarded, and can be treated with whatever method never jeopardizing the network QoS.

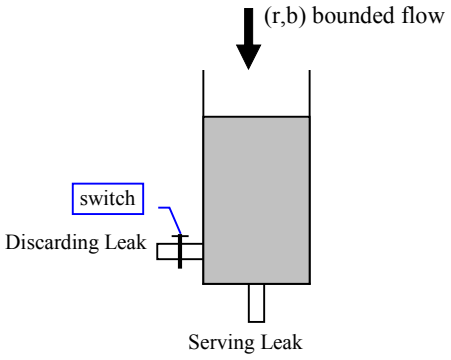


Fig. 4: Double-Leaks Bucket (Liquid Model)

Let C_1 and C_2 denote respectively the capacities of SL and DL. The control switch of DL works according to the quantity of the workload (represented by the height of

the water in the bucket and denoted by q). This function is called as *double threshold control function* (DTC function), as shown in Fig. 5, where 1 stands for the opened state of the DL switch (or water gate) and 0 stands for the closed state.

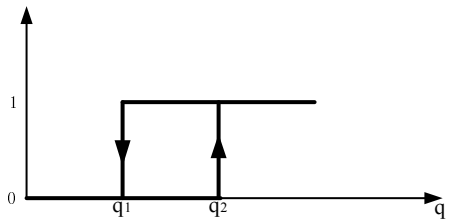


Fig. 5: Double Thresholds Control Function of the Switch

3.2 Service curve under (r,b)-bounded arrival stream.

We take the (r,b)-bounded arrival stream as a general source model. This is to say that the cumulative arrival curve is upper bounded by (r,b) [12], where r is the average arrival rate, and b is the burst. Denoting by $F_i(t)$ the function of the arrival curve, the workload arrived at any interval *[s, t]* is upper bounded by:

$$F_i(t) - F_i(s) \le b_i + r_i(t-s) \quad \forall 0 \le s \le t$$

This (r,b)-bounded source generates k units of workload in a time length no smaller than $l = \left(\frac{k-b}{r}\right)^+$ [12].

Assuming that the workload arrival bound is under the R-(m,k)-firm constraint. During the increment of the water height, the switch of Fig.4 remains closed until the height increases to q_2. Once it is opened at q_2, it remains opened state until the height reduces to q_1. When the switch is opened (control function's value is 1), the Discarding Leak throws out the water from the bucket, so that the water height will be effectively reduced to assure delay factor. During this procedure, it is obvious that no more than C2/(C1+C2) quantity of water can pass through DL, such that this attribute will be used to guarantee the (m,k) factor.

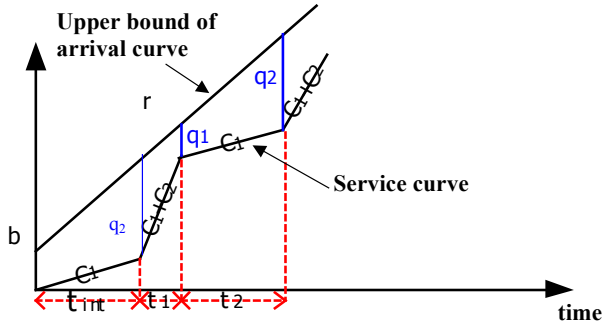


Fig. 6: Service curve evolution of DLB

Fig. 6 shows the evolution of the service curve under the critical cumulative arrival curve. Notice that SL is always on service unless the bucket is empty.

3.3 Sufficient condition for liquid model of DLB

After understanding the mechanism and its attribute to gurantee (m,k) factor and delay factor, we will propose the sufficient condition to configure DLB in order that deterministic guarantee can be achieved.

As shown in Fig.6, R-(m,k)-firm constraint is proposed in condition of congestion or overload, such that it is obvious that C1 < r. So the height of water grows higher and higher in the interval t_{int}, as the DL switch is closed initially. Once the height reaches q2, DL switch is opened due to the double threshold control function (Fig.5). In addition, we give C1+C2>r, so that the height of water goes lower and lower until the height decreases to q1 (in the interval t1). This procedure is iterated on until all arrival water has passed through either SL or DL.

Theorem 1:

If the liquid model of DLB is configured under following condition, then the R-(m,k)-firm constraint will be deterministically guaranteed.

Condition (1) : $C_1+C_2>r;\ \frac{C_1}{C_2} \geq \frac{m}{k-m} \Rightarrow C_1 \geq r\frac{m}{k}$;

Condition (2): if $b<q_2$, then $\max\left(\frac{q_2}{C_1}, \frac{q_2-q_1}{C_1+C_2}+\frac{q_1}{C_1}\right) < \Delta$

else $\max\left(\frac{q_2}{C_1}, \frac{b-q_1}{C_1+C_2}+\frac{q_1}{C_1}\right) < \Delta$

Proof:

Condition (1) ensures the (m,k) factor.

As the liquid model, water is an infinitesimal material (the unit could be atom, molecule, gram or ton, etc.). We configure C_1, and C_2 as $\frac{C_1}{C_2} \geq \frac{m}{k-m}$. Let $Q_{SL}(t)$ represent the throughput quantity through SL during]0,t], and $Q_{DL}(t)$ represent the throughput quantity through DL. It can be ensured that $\frac{Q_{SL}(t)}{Q_{DL}(t)} \geq \frac{C_1}{C_2} \geq \frac{m}{k-m}$, since SL is always on service unless the bucket is empty, while DL is only on service during the switch is open. Thus in any k units of water passed through SL and DL, there are at least *m* units passing through SL.

Condition (2) ensures the transmission delay factor of R-(m,k)-firm : when *k* units of water is affused into the DLB at time $t_a=\left(\frac{k-b}{r}\right)^+$. We calculate the maximum delay of a packet serviced by either SL or DL.

One case is when the switch of DL is open, and this case is further divided into two sub-cases: $b>q_2$ and $b<q_2$.

If $b>q_2$, the burst causes the maximum bucket load, then SL and DL service the workload together until the height decreases to q_1. Thereafter, SL service alone until the burst is finished. We know that results in the delay of the service for the burst is given by:

$$\frac{b-q_1}{C_1+C_2}+\frac{q_1}{C_1} \qquad (1)$$

In this case, if $b<q_2$, the height must decrease once the switch is open, so the maximum delay can be given by:

$$\frac{q_2-q_1}{C_1+C_2}+\frac{q_1}{C_1} \qquad (2)$$

Another case is that the height is so near to q_2, but the switch is still closed. SL will service alone the workload until empty the bucket. This case results in the delay of service, given by:

$$\frac{q_2}{C_1} \qquad (3)$$

It must assure that in any case the maximum delay is no more than $\triangle$, so that we give the condition (2).

End of proof.

In a concrete system, the source has its attributes of arrival curve upper bounded by (*r,b*) and R-(m,k)-firm requirement, so in the analysis, these parameters are regarded as the known parameters. Moreover, we can get the available bandwidth, so C_1 should be configured under the available bandwidth. Based on these known parameters we

should constitute one DLB (configure C_2, q_1, q_2) to cope with this flow for providing the deterministic R-(m,k)-firm guarantee.

3.4 Numeric application of liquid DLB

To show the numerical application, audio-CBR streams' parameters are taken into consideration. For example, one flow generates 2Mbps of average arrival rate with 6kbit of burst, such that the flow is bounded by (r,b)=(2Mbps, 6kbit). In this example, we assume that such the stream is under R-(3,5)-firm constraint and with $\triangle$=20ms as the per-flow deadline.

Assuming that at admission control, only 1.6Mbps bandwidth is available. Obviously, it is not possible to guarantee the deadline of all the packets, as congestions are unavoidable. We set $C_1+C_2 = 1.25r = 2.5$Mbps, so that the queue length can be effectively reduced in case of congestion. Then, DLB is implemented and configured as C_1=1.5Mbit/s; and C_2=1Mbit/s according to $C_1 = \frac{m}{k-m} C_2$; moreover, q_1 and q_2 are configured as that q_1=6kbit, q_2=12kbit. With this configured DLB, the maximum delay can be calculated by formula (2) as 8ms. So we can guarantee $t_{delay} \leq 8\text{ms} \leq \triangle = 20\text{ms}$. While, for guaranteeing all packets under delay of 20ms, it requires $r+b/t_{delay}$= 2.3Mbit/s of bandwidth [12]. With DLB, R-(m,k)-firm constraint is guaranteed in providing 8ms delay, but it requires only 1.5Mbit/s of bandwidth. Intuitively, in this example, DLB can still work under smaller bandwidth, so DLB is robust with smaller bandwidth.

3.5 Packet model of Double-Leaks Bucket

In the packet switching network the information are encapsulated as packets, and the non-preemption is widely employed. So we now develop the DLB model according to the granularity of the packets and under the discrete time. Afterwards, the given R-(m,k)-firm is oriented the number of packets. Fig.7 depicts the mechanism. Two new parts are added, named *temporary vessel buckets* (TVB) for DL and SL, respectively. They take an entire packet from the DLB after the service of the current packet in themselves. TVB of the DL can only get the next during the switch is still in opened state.

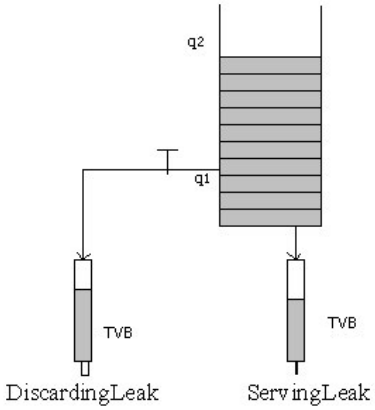


Fig. 7: Packet Model of DLB

The switch here is also controlled by the DTC function. Afterwards, for the packet model, the values of q_1 and q_2 are no longer the quantity of the water, but the number of packets; to represent the quantity of workload we use q_1S or q_2S, where S denotes the size of the packet (in unit of bit or byte).

3.6 Sufficient condition of DLB in packet model

Due to the packet granularity, the sufficient condition will be more complex than that of the DLB liquid model, which is given in the following theorem.

Theorem 2:
The sufficient condition for guaranteeing R-(m,k)-firm constraint of the packet model can be given as following:

(1) $q_1 \geq \frac{C_1}{C_2} \geq \frac{m}{k-m}$

(2) If $b<q_2$, then $\max\left(\frac{q_2-1}{C_1}S, \left(\frac{q_2-q_1}{C_1+C_2}+\frac{q_1}{C_1}\right)S\right) \leq \Delta$

else $\max\left(\frac{q_2-1}{C_1}S, \left(\frac{b-q_1}{C_1+C_2}+\frac{q_1}{C_1}\right)S\right) \leq \Delta$

Proof:
Condition (1) ensures the (m,k) factor.

As mentioned in the liquid model, it should be provided that $\frac{C_1}{C_2} \geq \frac{m}{k-m}$ as in the liquid model. Furthermore, we must take into account the granularity of the packet. In case that TVB of DL has just taken one packet when the bucket height is q_1+1, then the switch will be closed. Therefore, the service time of TVB of DL will continue

$\frac{S}{C_2}$, and SL will be on service at least $\frac{q_1}{C_1}S$. The service process should be that SL is always on service during DL does, then $\frac{q_1}{C_1} > \frac{1}{C_2}$. This deduces that $q_1 \geq \frac{C_1}{C_2} \geq \frac{m}{k-m}$. We set q_1 as the minimum value such as: $q_1 = \left\lceil \frac{m}{k-m} \right\rceil$, and it is clear that $q_2 > q_1$. So we can choose one suitable value for q_2, which is neither too much big nor causes the switch rotate too frequently.

Condition (2) is derived with the same stragety as that for liquid model to guarantee delay factor.

End of proof.

3.7 Numeric application of packet DLB model

Let's consider the same concrete example as in the liquid model, the packet size is S=6kbit, then we can configure this DLB as C_1=1.44Mbit/s, C_2=0.96Mbit/s, q_1=2, q_2=5. With this configured DLB, R-(m,k)-firm is guaranteed $t_{delay} \leq \frac{q_2 - 1}{C_1}S$ =10ms $\leq \triangle$ = 20ms.

4 Comparison with other real-time constraint relaxation strategies

The R-(m,k)-firm scheme is a strategy to relax the too tight (and unnecessary) hard real-time constraint. Similarly, Pinwheel model [13], frame-based model [14] and Virtual window constrained model [11] can also be considered as other constraint relaxation strategies. In what follows we will discuss on those models and show that R-(m,k)-firm is a general model which can include the other ones.

4.1 Limitation of utilization gain when deterministic (m,k)-firm guarantee is required

The initial motivation of (m,k)-firm constraint is that the deadlines of at least *m* out of any *k* consecutive packets must be guaranteed. Similar to (m,k)-firm constraint, Dynamic Window-Constraint Scheduling (DWCS) [10] can accept a certain deadline misses such that *x* is the maximum acceptable packet loss in a fixed given window *y* packets.

(*m,k*)-firm and DWCS constraints just take into account the quantity of the deadline met or miss, whereas a stream is defined with several other parameters, such as *transmission time, period and dropping rate*. This unilateralism of analysis results in

bad utilisation of resource, and this poor utilization factor causes the considerable over-provisioning of the system. Again, this over-provisioning causes the (*m,k*)-firm or its similar systems (e.g. DWCS) to loss their practical interest. Those facts have been shown in [7, 8, 9].

In [8], one theorem is proposed to show the utilization factor of a steam set serviced by multiprocessor under window-constraint. The result is very pessimistic, such that the system falls into failure state (no schedulable) even in arbitrary low utilization. The theorem is given as follows:

> For an arbitrary non-negative real number u and a natural number g, there exists a unit-size Window-Constrained stream set Γ, such that the aggregate utilization rate of Γ is less than or equal to u, and Γ is not schedulable by DWCS [10] on g processors.

In fact, such a theorem has also been given for (m,k)-firm constraint in [9]. Such as that:

> Under an arbitrary low utilisation, there is always a stream set Γ, which causes the system violate the (m,k)-firm constraint.

As mentioned above, providing deterministic (*m,k*)-firm guarantee can oblige resource reservation according to (*k, k*)-firm. This is to say that (*m,k*)-firm losses its practical interest for reducing the necessary resource reservation.

4.2 Complexity fatality

More generally, in [7, 8], it has been proven that:

> The schedulability of a stream set under (m,k)-firm or DWCS is a problem of NP-hard in strong sense.

4.3 Other relaxed constraint 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 try to relax the deadlines to gain higher utilization factor, such as frame-based model [14], Pin-wheel scheduling [13] and virtual deadline window scheduling [11], etc.

The frame-based model defines a set of tasks, which is to execute within each frame (time window) and is to complete before the end of frame. The problem is to schedule the task set in a single frame with deadline D.

Frame-based model assumes that all task 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 tasks' instances size is inferior or equal to the frame size, any no-idle scheduling can treat with them without any constraint violation. The scheduling in a frame can be repeated to do with infinite task instances.

Note that frame-based model removes the deadlines of instances to provide a scheduling in the same frame size for all tasks. However, for current diverse applications, tasks demand different frame size. Which makes the frame-based model little interest 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 task. 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 $\{1, 2, 3, \ldots, n\}$ 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 tasks 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 [11] proposed a relaxed window constraint to gain utilisation. It allows task instances to be serviced after their deadlines, as long as it can guarantee a minimum fraction of service to a task 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 tasks 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.

In one word, R-(m,k)-firm synthesized all the above mentioned relaxation strategies to define a more general scheme for general task set. This scheme can be deterministically guarantee using DLB under the sufficient condition (theorem 1 and 2) proposed in section 3.

In the next subsection, we will shown the advantage gained by R-(m,k)-firm scheme. Simultaneously, it must be noticed that R-(m,k)-firm constraint can be well guaranteed even with the traditional simple scheduling policy, such as Rate Monotonic policy.

4.4 Comparison by simulation

In this subsection, we give a scenario to show that R-(m,k)-firm constraint can significantly gain the utilisation factor compared with (m,k)-firm constraint and DWCS constraint.

The scenario consists in four flows $\{\tau_1, \tau_2, \tau_3, \tau_4\}$, each of which has R-$(m_i,k_i)$-firm constraint. We will simulate for a strict delay factor (i.e., Δ=0) to show the high gain of utilisation.

The stream set is scheduled under non pre-emptive fixed priority scheduling, and the priority is indicated by the index. Then, the schedule trace is shown in Fig.8.

Notice that R-(m,k)-firm constraints are well guaranteed for either sliding window or no-overlapping window with zero delay (Δ=0). Such that, for any stream, in any

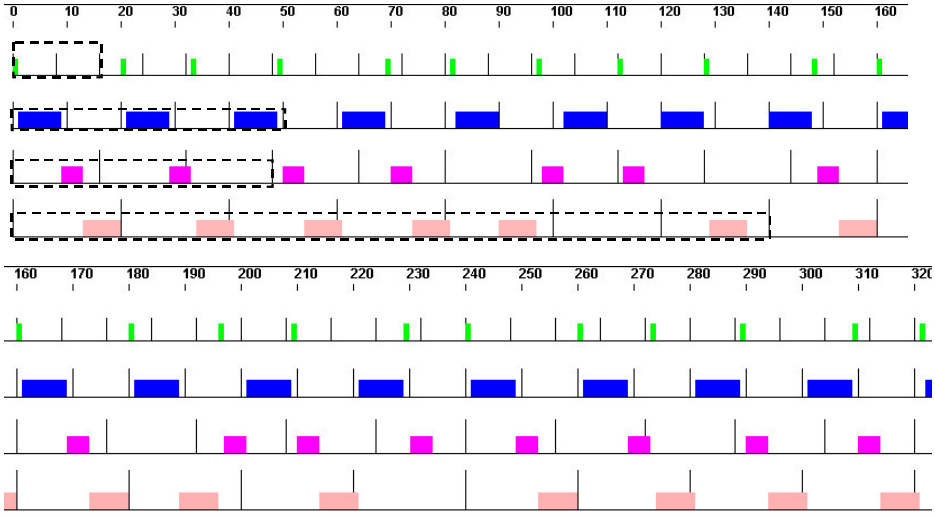
window of size $k_i p_i$, there are at least m_i instances executed. Dashed window is marked in Fig. 8, which can help the readers to verify this.

	Period (ms)	Execution time (ms)	R-(m,k)-firm
τ_1	8	1	(1,2)
τ_2	10	8	(2,4)
τ_3	16	4	(2,3)
τ_4	20	7	(5,7)
Utilisation		88% with Δ=0	

Table 1: R-(m,k)-firm simulation scenario

It is also easy to discover that a lot of missed deadlines exist during the scheduling trace for τ_3 and τ_4; caused by this, all deadline schemes (DWCS and (m,k)-firm constraint) will be violated. However, R-(m,k)-firm constraint replaced the per-instance deadline by the delay of group of instances, such that a high utilisation is gained (88%). Note that more gain of utilisation can be achieved with the no-zero delay factor.

Moreover, this scenario dealt with the periodic task set, which makes it different from the numerical applications given in section 3. The numerical applications given in section 3 are applications of DLB for the tasks upper bounded by (r,b), while periodic task has less burst. Again, none of the other deadline schemes can achieve the same gain in case of either this scenario or the numerical applications in section 3.



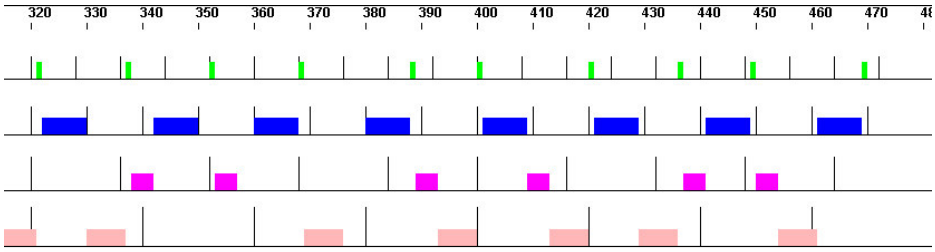


Fig.8: Scheduling trace of Table

5. Performance comparison

As shown above, DLB can provide deterministic *R-(m,k)-firm* guarantee to *periodic and (r,b)-bounded* tasks (or flows). To show the performance of DLB faced to more general flows, we simulated DLB as well as TD and RED in case of a Poisson flow of the rate $\lambda=1$ packet per time unit, with equal packet size. Assume also that the source generating this flow can tolerate up to 33% of loss. To simulate their behaviors during overload period, the server capacity is set to be less than the average arrival rate of the source. For equal comparison, the same queue size and server capacity are given to DLB, TD and RED (parameters are shown in Fig 9).

DLB pattern (q_1=3, q_2 = 6, queue size = 9, C_1 = 0.8, C_2 = 0.4)
01101101101101101101101101101101101101111111111111111111011011011011011011011011111111110110110110110110110110110110110110110111111111111111111111111111111111111101101101101101101101101101101101101101
TD pattern (queue size = 9, C = 0.8)
10110111110000110001111111111111111111111111111011101010010111111111011111101001111110101001010000111110011111111111111111111111111111111111111101110110011010001111111011111101111110
RED pattern (w_q = 0.2, max_p = 0.34, min_{th} = 3, max_{th} = 6, queue size = 9, C = 0.8)
11101001000011111111111111111111111111111111101011111111111001100111110111111111111111111111111111111011001011101111101111111111111111111101000011110001100010101111111111111111111101010

Fig. 9: Loss Packet patterns of DLB, TD and RED

Figure 9 also shows a sample sequence of the served and dropped packets, denoted by 1 and 0, respectively. As seen, DLB avoids consecutive losses, while TD and RED do not. Additionally, DLB performs also well in terms of queue length and queuing delay, such as shown in Table 2.

	DLB	RED	TD
Mean queue length	4.7	4.3	9.0
Mean delay (s)	4.8	4.5	8.7
Drop rate	22%	21%	20%

Table 2: Comparison of DLB, RED and DT.

6. Perspective to Internet QoS

DLB can be easily implemented in place of the traditional leaky bucket. So this approach can find many applications in the current network to provide the adaptive controllable QoS according to the R-(m,k)-firm constraint.

In Diffserv, DLB can be implemented in each priority dropping queues, such as AF1, AF2, AF3 and AF4 in Fig.10. The classifier puts a stream into a certain priority-dropping queue according to its requirement (if it can be admitted). On the other hand, DLB can also act as the classifier such that packets passing through SL will be entered in EF queue. The packets passing through the DL will be entered into best effort queue. Then, a stream is split and real-time constraint can be guaranteed by EF queue. Obviously, in this way EF queue can admit more streams than without dropping.

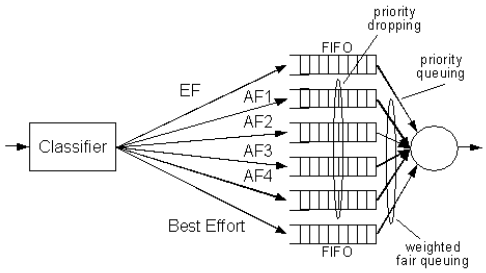


Fig. 10: DLB implemented in DiffServ

In MPLS model, when packets pass through the previous router, they will be marked in the label if the switch of DLB was open. This ensures that in this slice of stream, no more packets will be dropped in the next hop (router). Nevertheless, the dropping can still be done in the slices where the label is not marked.

In one word, DLB is very flexible and could be compatible with almost all the current network schemes.

7 Conclusion

Since the current networks often falls into congestion caused by the overload otherwise the over-provisioning is not always possible, graceful degradation of Quality of

Service in networks is necessary for efficiently supporting the packet loss tolerant real-time applications such as VoIP, VoD, etc. Selectively discarding packets according to the (m,k)-firm model during overload periods is the key issue of our approach.

In this paper, two main contributions can be found. One is the novel real-time QoS constraint, named as Relaxed (m,k)-firm constraint. Under this R-(m,k)-firm constraint, long consecutive loss of packets can be avoided even with a high bandwidth utilization, such that it is suitable for divers multimedia applications. This novel real-time constraint replaces deadlines for each packet by a delay factor of a packet group, moreover, it orients to the more general (b,r)-bounded streams, including periodic and sporadic ones. Another contribution is that one dynamic mechanism, called Double Leaks Bucket, has been proposed to deterministically guarantee the R-(m,k)-firm constraint.

The comparison with other schemes showed the generality of R-(m,k)-firm constraint in contrast with DWCS, Frame-based Model, Pinwheel Model, and Virtual Deadline Scheduling model. Furthermore, Simulation scenarios showed how R-(m,k)-firm constraint increases the resource utilization by replacing deadlines of packets with the delay of group of packets. Comparisons among the packet dropping mechanisms showed the advantage of avoiding consecutive drops with DLB in contrast of RED and TD. Finally, the discussion of perspective showed the wide adaptation in the current Internet.

References

[1] El-Gendy, M.A., A. Bose, K.G. Shin, "Evolution of the Internet QoS and support for soft real-time applications", *proceedings of the IEEE*, Vol.91, No.7, pp1086-1104, July 2003.

[2] Floyd, S., and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance". In *ACM/IEEE Transactions on Networking*g, 3(1), August 1993.

[3] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m,k)-firm deadlines", IEEE Transactions on Computers, 44(4), pp1443–1451, Dec.1995.

[4] F. Wang and P. Mohapatra, "Using differentiated services to support Internet telephony", Computer communications, Vol.24, Issue18, pp1846-1854, Dec. 2001.

[5] Striegel A., G. Manimaran, "Dynamic Class-Based Queue management for scalable media servers", *Journal of systems and software,* vol.66, No.2, pp.119-128, May 2003.

[6] Koubâa, A., Song, Y.Q., "Loss-Tolerant QoS using Firm Constraints in Guaranteed Rate Networks", *10th IEEE Real-Time and Embedded Technology and Applications (RTAS'2004)*, Toronto (Canada) 25-28 May 2004.

[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] 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.

[9] 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.

[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] 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.

[12] Le Boudec, J.Y. and P. Thiran, *Network Calculus: A Theory of Deterministic Queueing Systems For The Internet*, Online Version of the Book of Springer Verlag – LNCS 2050, July 2000

[13] 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.

[14] 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.

[15] A. Koubâa, Y.Q. Song, "Evaluation and improvement of response time bounds for real-time applications under non pre-emptive fixed priority scheduling", *International Journal of Production Research*, Vol.42, No.14, pp2899-2913, Taylor & Francis group, July 2004.

[16] N. Jia, E. Hyon, Y. Song, "Ordonnancement sous contraintes (m,k)-firm et combinatoire des mots", RTS'2005, Paris (France), April 2005.