Bandwidth and Latency Analysis of Modified Deficit Round Robin Scheduling Algorithms

Luciano Lenzini
Enzo Mingozzi
Giovanni Stea

Dipartimento di Ingegneria dell’Informazione, University of Pisa, Italy
Via Diotisalvi 2, 56122 Pisa, Italy – Ph. +39 050 2217599
{l.lenzini, e.mingozzi, g.stea}@iet.unipi.it

ABSTRACT
Deficit Round Robin (DRR) is a scheduling algorithm which provides fair queuing at O(1) complexity. However, due to its round robin structure, its latency properties are not adequate for latency-critical applications, such as voice. For this reason, router manufacturers implement variants of the DRR algorithm which guarantee lower latencies to one (or a subset of) queue(s). In this paper we evaluate the performance of two such variants, both of which are known as Modified Deficit Round Robin, currently implemented in commercial routers. The comparison is carried out analytically, by deriving the latency and bandwidth sharing properties of both algorithms, and by simulation.

Categories and Subject Descriptors

General Terms
Algorithms, Performance.

Keywords

1. INTRODUCTION
Multi-service packet networks are required to carry traffic belonging to different applications, such as e-mail or file transfer, which do not require pre-specified service guarantees, and real-time video or telephony, which instead require service guarantees. The best-effort service model, though suitable for the first type of applications, is not amenable for applications of the other type. Therefore, multi-service packet networks need to enable Quality of Service (QoS) provisioning. A key component for QoS enabling networks is the scheduling algorithm, selecting which next packet to transmit, and when, on the basis of some expected performance. During the last decade, this research area has been widely investigated, as proved by the abundance of existing literature [2-18]. Different scheduling algorithms have been devised, which exhibit different fairness and latency properties at different worst-case per-packet complexity. Among those, Deficit Round Robin (DRR, [5]) achieves O(1) per-packet complexity, thus being amenable to high-speed implementations. However, its latency guarantees are looser than those of timestamp-based schedulers (like Packet Generalized Processor Sharing, PGPS, [3]). In fact, a time-sensitive packet arriving at the head of a queue may have to wait for every other (possibly best-effort) backlogged queue to be served before getting a service opportunity. Although such queuing delay is bounded, it might introduce a significant jitter, which is not tolerated by some real-time application (e.g., voice). In order to solve this problem – while preserving the original DRR design simplicity – two different approaches have been proposed. On the one hand, implementation techniques that reorder the sequence of packet transmissions within a round, so as to reduce latency and enhance fairness, have been proposed ([11], [13], [20]). On the other hand, variants of the basic DRR algorithm that prioritize one (or a subset of) queue(s), so as to guarantee a reduced latency, have been devised (e.g., DRR+ [5], and DRR++ [21]). Two such variants, both of which go under the name of Modified Deficit Round Robin (MDRR) have been implemented in commercial routers [18],[19]. The first one, hereafter called C-MDDR, is currently implemented in Cisco 12000 routers. Besides “standard” queues, which are serviced according to DRR, C-MDDR allows the definition of a low latency queue, which is given priority over the set of standard queues, and is mainly intended to support voice traffic. The second one, hereafter called J-MDDR, is implemented in Juniper M-Series routers. In J-MDDR, three priority levels are defined: low, high and strictly-high. However, the scheduling mechanisms used to enforce the various priorities in C-MDDR and J-MDDR are different. The contribution of this paper is a comparative performance analysis of the C-MDDR and J-MDDR scheduling algorithm. The analysis is carried by i) evaluating their latency and bandwidth sharing guarantees, and ii) simulating both algorithms with realistic traffic.

The rest of the paper is organized as follows. Section 2 provides a background on the DRR algorithm. We describe both C-MDDR and J-MDDR in Section 3. In Section 4, the bandwidth guarantees of both algorithms are investigated, whereas their latency properties are derived in Section 5. Simulation results are shown in Section 6. Finally, conclusions are drawn in Section 7.

2. BACKGROUND
C-MDDR and J-MDDR can be best described once the basic mechanism of Deficit Round Robin is well understood. For this reason, we describe the latter first.

Deficit Round Robin is a variation of Weighted Round-Robin
(WRR) that allows flows with variable packet lengths to share the link bandwidth fairly. Each queue $i$ is characterized by a quantum of $\phi_i$ bits, which measures the quantity of packets that queue $i$ should ideally transmit during a round, and by a deficit variable $\Delta_i$. When a backlogged queue is serviced, it is allowed to transmit a burst of packets of an overall length not exceeding $\phi_i + \Delta_i$. When a flow is not able to send a packet in a round because the packet is too large, the number of bits which could not be used for transmission in the current round is saved into the queue’s deficit variable. This deficit is then made available to the same queue in the subsequent round. More specifically, the deficit variable is managed as follows:

- reset to zero when the queue is not backlogged;
- increased by $\phi_i$ when the queue is selected for service during a round;
- decreased by the packet length when a packet is transmitted.

Let $L_i$ be the maximum length of a packet for queue $i$ (measured in bits). In [5] the following inequality has been proved to hold right after a queue has been serviced during a round:

$$0 \leq \Delta_i < L_i$$

which means that a queue’s deficit never reaches the maximum packet length for that queue.

In DRR, a FIFO list, called the active list, stores the references to the backlogged queue. When an idle queue becomes backlogged, its reference is added at the tail of the list. Cyclically, if the list is not empty, the queue which is at the head of the list is extracted and serviced. After being serviced, if still backlogged, the queue is added at the tail of the list. It is worth noting that:

- extracting and inserting queues in a FIFO list are $O(1)$ operations;
- since backlogged queues are demoted to the tail of the list after reaching the head of the list and receiving service, the relative service order of any two backlogged queues is preserved through the various rounds.

It has been proved in [5] that the following inequality must hold for DRR to exhibit $O(1)$ complexity:

$$\forall i, \phi_i \geq L_i$$

Inequality (2) states that each queue’s quantum must be large enough to contain the maximum length packet for that flow. If some queues violate (2), it is possible that a queue which is selected for transmission (i.e. dequeued from the head of the list), though backlogged, cannot transmit a packet during the current round. Nevertheless, its deficit variable must be updated and the queue must be queued back at the tail of the list. In a worst case, this can happen as many consecutive times as the number of queues violating (2). Therefore, if (2) does not hold, the number of operations needed to transmit a packet in the worst case can be as large as $O(N)$ [14]. Let us assume that DRR is used to schedule packets from $N$ queues on a link whose capacity is $C$. Let

$$F = \sum_{i=1}^{N} \phi_i$$

be the frame length, i.e. the number of bits that should ideally be transmitted from the queues during a round if all were backlogged. In DRR the minimum guaranteed rate of queue $i$ is:

$$R_{i}^{\text{DRR}} = \frac{\phi_i}{F} \cdot C$$

As far as delay guarantees are concerned, two metrics are commonly considered for packet schedulers: the latency and the start-up latency. The former, introduced in [2], is related to the maximum lag that a packet can experience with respect to the case of a dedicated reference server. As such, it can be used to compute tight worst-case delay bounds when the arrivals are constrained (e.g. by a leaky bucket). The latter, also called scheduling delay, is the maximum time that a head-of-line packet can be delayed due to scheduling decisions. For DRR, a tight bound on both metrics has been computed in [20]. The latency of DRR is:

$$\mathcal{O}_{i}^{\text{DRR}} = \frac{1}{C} \left[ F - \phi_i \left(1 + \frac{1}{N} \sum_{j=1}^{N} L_j \right) \right]$$

whereas the start-up latency of DRR is:

$$S_{i}^{\text{DRR}} = \frac{1}{C} \left[ F - \phi_i \left(1 + \sum_{j=1}^{N} L_j \right) \right]$$

Note that, if $\phi_i \geq L_i$, then (6) becomes:

$$S_{i}^{\text{DRR}} = \frac{F - \phi_i + \sum_{j=1}^{N} L_j}{C}$$

Expressions (5) and (6) show that both the latency and start-up latency depend on the number of active queues, since they include the frame length $F$ and the sum of the maximum packet length of all queues. Furthermore, they cannot be effectively tuned by varying queue $i$’s quantum, due to the presence of other constant terms. Therefore, latency-critical applications (like voice) can suffer from potentially high queuing delays, regardless of the allocated quantum. In the next section, we show how C-MDRR and J-MDRR define additional scheduling mechanisms in order to mitigate this problem.

3. DESCRIPTION OF C-MDRR AND J-MDRR

3.1. C-MDRR

In C-MDRR, standard queues are serviced according to DRR. Furthermore, a single Low Latency Queue (LLQ), to which higher priority is given, can be defined. In C-MDRR, the LLQ can work either in strict or alternate priority mode. Both priority modes are described below.

3.1.1 Strict Priority

In the strict priority mode, the LLQ is always serviced in exhaustive, non preemptive priority mode. The other queues are serviced cyclically, as in DRR, whenever the LLQ is empty. A standard queue can thus have its service turn interrupted by the arrival of a packet in the LLQ. The behavior of strict priority C-MDRR is illustrated in Figure 1, left.
3.1.2 Alternate Priority

In the alternate priority mode, the LLQ is assigned a quantum. Whenever non-empty, the LLQ is serviced for its whole quantum every second service turn. Thus, if \( N \) standard queues (numbered 1 to \( N \)) are defined, and all queues (including the LLQ) are backlogged, the service order in a round is: LLQ, 1, LLQ, 2, ..., LLQ, \( N \). The behavior of alternate priority C-MDRR is illustrated in Figure 1, right.

![Figure 1 – strict-priority (left) and alternate-priority (right) in C-MDRR](image)

3.2. J-MDRR

In J-MDRR, a queue can have a low, high or strictly-high priority. A strictly-high priority queue is serviced whenever it is non-empty, like the LLQ in strict priority mode in C-MDRR. On the other hand, high and low priority queues are stored in two different active lists, as shown in Figure 2.

![Figure 2 – J-MDRR](image)

Both high and low priority queues are serviced for a quantum on each round, and they carry on their deficit to the subsequent round if they are still backlogged. In a round, the active list of high-priority queues is serviced first, until either it is empty or all high-priority queues have been serviced for their quantum. Low-priority queues are serviced next. However, unlike C-MDRR, low and high priority queues do not transmit a quantum worth of packets in a single burst. Instead, they transmit one packet at a time: if queue \( j \) can transmit more than one packet in the round, it is queued back at the tail of the respective active list after transmitting each single packet. Thus, a queue can be serviced more than once per round. This technique has been shown to effectively reduce burstiness in [20]. Note that an arbitrary number of high priority queues can be defined in J-MDRR, whereas only a single strictly-high priority queue can be defined.

4. BANDWIDTH GUARANTEES

In this section, we analyze the bandwidth sharing properties of C-MDRR and J-MDRR. More specifically, we derive expressions for the minimum guaranteed rate of a queue under both disciplines. Furthermore, we investigate how the “tuning knobs” provided by each algorithm affect the bandwidth sharing capabilities. Hereafter, we use \( F \) to denote the sum of the quanta of the standard queues in C-MDRR, and of the high- and low-priority queues in J-MDRR.

4.1. Bandwidth guarantees of C-MDRR

Let us assume that C-MDRR is used to schedule packets from \( N \) standard queues and an LLQ.

4.1.1 Strict Priority Mode

When C-MDRR works in strict priority mode, the LLQ has access to the full channel capacity. This implies that there is no means to control the amount of bandwidth that the LLQ can get, and standard queues can actually be starved. Therefore, standard queues cannot be guaranteed a minimum service rate (i.e., \( R^s_i = 0 \)), unless constraints on the LLQ traffic are enforced.

4.1.2 Alternate Priority Mode

When C-MDRR works in alternate priority mode, the rate at which the LLQ is serviced is determined by the ratio of the quanta. When all flows are backlogged, the rate of the LLQ is:

\[
R^a_{LLQ} = \frac{N \cdot \phi_{LLQ}}{\sum_{i=1}^{N} \phi_i + N \cdot \phi_{LLQ}} \cdot C = \frac{N \cdot \phi_{LLQ}}{F + N \cdot \phi_{LLQ}} \cdot C, \quad (8)
\]

whereas for a standard queue it is:

\[
R^a_i = \frac{\phi_i}{\sum_{i=1}^{N} \phi_i + N \cdot \phi_{LLQ}} \cdot C = \frac{\phi_i}{F + N \cdot \phi_{LLQ}} \cdot C \quad (9)
\]

Note, however, that \( R^a_i \) is the minimum guaranteed rate for a standard queue — \( R^a_{LLQ} \) is not the minimum guaranteed rate for the LLQ. In fact, the latter is not achieved when all queues are backlogged. Instead, it is achieved when a single standard queue, whose quantum \( \phi_{max} \) is the maximum among all standard queues, is backlogged, and it is equal to:

\[
R^a_i = \frac{\phi_{LLQ}}{\phi_{max} + \phi_{LLQ}} \cdot C \leq R^a_{LLQ} \quad (10)
\]

Note that \( R^a_{LLQ} = \phi_{LLQ} \) only when all the standard queues have the same quantum. Therefore, the minimum rate of the LLQ is achieved when some standard queues are idle, i.e., under load conditions different than those used for assessing the minimum rate of standard queues. As a remarkable consequence, this constrains the maximum link utilization. Assume for example that C-MDRR is used to schedule a set of three flows on a 100Mbps link. The flow mapped to the LLQ requires a minimum guaranteed rate of 30Mbps, while the other two flows, mapped to standard queues, require 10 and 50 Mbps respectively. The rate requirements can be described by the following set of inequalities:
Although the overall requirements are 90% of the link capacity, it can be easily verified that there is no way of selecting quanta so that all inequalities in (11) are satisfied.

4.1.3 Problems with quantum selection

In this subsection we describe problems connected to the quantum selection in C-MDRR. To this purpose, we refer to the C-MDRR implementation described in [18], in which queues are assigned an integer weight, ranging from 1 to 2048. The way a quantum is computed starting from a weight is the following:

\[
\phi = M + (w - 1) \cdot \Delta
\]  

(12)

where \( M \) is the MTU on the link and \( \Delta \) is a fixed increment (512 bytes in Cisco GSR routers). Thus, a queue with the lowest weight (equal to 1) is allowed to send an MTU on each round – which is required for C-MDRR to work at \( O(1) \) worst-case complexity. Note that, if strict priority is employed, the weight assigned to the LLQ is not relevant. Therefore, the quantum is a non-linear function of the weight, the non-linearity being given by the offset \( M \) in (12). Now, \( M \) can be (and actually is, in practical cases) much larger than \( \Delta \). For example, in a token ring link, we have \( M = 4500 \) bytes, so that two standard queues with weight 1 and 2 respectively get a bandwidth ratio which is 0.9:1. Thus, it is quite hard (if not impossible) to allocate bandwidth in a purely weighted way.

If alternate priority is employed, the problem of allocating a suitable weight to the LLQ has to be taken into consideration as well. Let us denote with \( f_{LLQ} \) the link capacity share required for the LLQ (determined in whatever meaningful way). By merging (10) with (12), after some straightforward computation, we obtain a lower bound for the LLQ weight:

\[
w_{LLQ} \geq 1 + \frac{M \cdot \left(2f_{LLQ} - 1\right) + \left(w_{\text{max}} - 1\right) \cdot \Delta \cdot f_{LLQ}}{\Delta \cdot \left(1 - f_{LLQ}\right)}
\]  

(13)

Note that, as \( f_{LLQ} \) approaches 1, the weight required in order that the LLQ achieve the required capacity goes to infinite. The fraction at the right hand side of (13) can take negative values, which means that a quantum of less than an MTU would be sufficient to grant the required bandwidth to the LLQ. However, weights must be positive integers. We can thus re-write (13) as:

\[
w_{LLQ} \geq 1 + \left[ \frac{M \cdot \left(2f_{LLQ} - 1\right) + \left(w_{\text{max}} - 1\right) \cdot \Delta \cdot f_{LLQ}}{\Delta \cdot \left(1 - f_{LLQ}\right)} \right]^{\frac{1}{x}}
\]  

(14)

where \( \left[x\right] = \max\{0, x\} \). Weight quantization according to (14) may result in significant overprovisioning for the LLQ. We provide an evidence of the above problem with a simple example. Let us assume that we have 10 standard queues, 5 of which with weight 2 and five with weight 1. The MTU is 1500 bytes, as in Ethernet. We compute the level of overprovisioning implied by (14) as a function of the required bandwidth share for the LLQ. Figure 3 clearly shows that, when \( f_{LLQ} \) is low (below 0.3), the amount of overprovisioning with respect to (13) can be very high, (e.g. about 350% for \( f_{LLQ} = 0.1 \)). Furthermore, the overprovisioning given to the LLQ can vary according to a nasty sawtooth behavior, as shown in Figure 4.

In order to reduce the overprovisioning for low values of \( f_{LLQ} \), as well as the effect of weight quantization, it could be envisaged to scale upwards the weight for the standard queues. For example, if we multiply each weight by 10 in the above example, the overprovisioning is remarkably reduced (see Figure 5 and Figure 6). However, this comes at the expense of increasing all quanta. Accordingly, transmission from the standard queues is likely to occur in large bursts. Furthermore, as proved in the next subsection, this affects the latency for both the standard queues and the LLQ.

4.2. Bandwidth sharing properties of J-MDRR

If a strictly-high priority queue exist in J-MDRR, it has access to the full channel capacity. In that case, unless constraints on its traffic are enforced, there is no means to guarantee that high and low priority queues are not starved. If no strictly-high priority queue is defined, both high and low priority queues have a minimum guaranteed rate. Assuming that J-MDRR is used to schedule \( N \) queues, having either high or low priority, their minimum guaranteed rate is exactly the same of DRR (4), i.e. is still determined by the ratio of its quantum to the frame length. Thus, the bandwidth guarantees in J-MDRR are the same for both high and low-priority queues.

4.2.1 Problems with quantum selection

In J-MDRR, each queue is assigned a given rate. For each queue, the deficit is replenished 5000 times per second. Therefore, the quantum of a queue is computed as:

\[
\phi = r_{i}/5000
\]

This implies that a queue with a low rate might have a quantum which is smaller than its maximum packet size, and therefore be unable to transmit any packet in a round, though backlogged. Therefore, J-MDRR is not guaranteed to work at \( O(1) \) per-packet complexity.
5. LATENCY ANALYSIS

In this section, we derive expressions for the latency and start-up latency of all types of queue in C-MDRR and J-MDRR.

5.1. C-MDRR

5.1.1 Strict Priority Mode

When C-MDRR works in strict priority mode, an LLQ packet can only be delayed by at most one packet from a standard queue. Therefore, the LLQ is guaranteed a service curve of the latency-rate type, with a rate

\[ S_{LLQ} = \frac{1}{\max \{T_i\} + \frac{L_{LLQ}}{C}} \]  

(15)

Note that \( S_{LLQ} \) is the lowest latency that can be guaranteed by a scheduler managing more than one queue. It can also be straightforwardly proved that \( S_{LLQ} \) is also the start-up latency of the LLQ.

On the other hand, it is well known that lower priority (i.e., standard) queues can actually be starved when strict priority scheduling is in place. Therefore, standard queues cannot be guaranteed a minimum service rate (i.e., \( R_i^S = 0 \)) or a maximum latency (i.e., \( \Theta_i^S = \infty \)), unless constraints on the LLQ traffic are enforced.

5.1.2 Alternate Priority Mode

When C-MDRR works in alternate priority mode, an LLQ packet can be delayed by a full quantum (plus possibly at most one maximum size packet of accumulated deficit). As far as latency is concerned, we can state the following results.

Theorem 1

The latency of the LLQ in alternate priority C-MDRR is:

\[ \Theta_{LLQ}^A = \frac{1}{C} \left[ \sum_{j=1}^{N} (\phi_j + L_j - 1) \cdot \phi_{MAX} + \frac{L_{LLQ} \cdot \phi_{MAX}}{\phi_{LLQ}} \right] \]  

(16)

where \( X = \{ j : 1 \leq j \leq N, \phi_j + L_{LLQ} \geq \phi_{MAX} \} \). Furthermore, for a standard queue \( i \), it is:

\[ \Theta_{i}^A = \frac{1}{C} \left[ \left( F + N \cdot \phi_{LLQ} - \phi_i \right) \cdot \left( 1 + L_j / \phi_i \right) + \sum_{j=1}^{N} L_j / L_{LLQ} \right] \]  

(17)

Proof: see Appendix.

Note that set \( X \) is a subset of \( \{1, ..., N\} \), and it is non-empty since it contains at least the flow with the maximum quantum. As a straightforward corollary, we obtain that the latency for the LLQ is upper bounded by:

\[ \Theta_{LLQ}^A \leq \frac{1}{C} \left[ \sum_{j=1}^{N} L_j / L_{LLQ} + 2 \cdot \phi_{MAX} \right] \]  

(18)

We observe that equality may actually hold in (18), e.g. if all standard queues have the same quantum and the LLQ has a quantum equal to the maximum packet size. By comparing (18) with (5), we observe that alternate priority C-MDRR is effective in reducing the latency of the LLQ (with respect to DRR), at the expense of increasing that of the standard queues. Note, however, that the latency of the LLQ still accounts for one maximum size packet from each standard queue, and hence still depends on the number of flows. Furthermore, the LLQ latency does not heavily depend on the LLQ quantum. Therefore, it is impossible to reduce the former by tuning the latter, as normally happens in rate-based schedulers. On the contrary, the latency for the standard queues is increased with respect to DRR. We show this with a numerical example in Figure 7. Suppose that \( N+1 \) queues are defined, all of which have a max. packet length equal to 1500 bytes. One queue has a quantum equal to 1500 bytes, and we vary the quantum of all the other \( N \) queues from 1500 to 4500 bytes. The same set of queues, with the same quantum, is scheduled in C-MDRR and DRR. In C-MDRR, the queue with the fixed quantum is selected as an alternate priority LLQ in C-MDRR. We plot the ratio of the latency in C-MDRR with respect to DRR as a function of the \( N \). The figure shows that a remarkable latency reduction is achieved for the LLQ in C-MDRR, especially as the number of queues gets higher. On the other hand, the latency for the standard queues in C-MDRR is remarkably increased.
Figure 7 – latency ratio with respect to DRR

Figure 8 – start-up latency ratio with respect to DRR

For alternate priority C-MDRR, the start-up latency of the LLQ is:

\[ S_{LQ}^A = \frac{1}{C} \max_{N \in \mathbb{Z}^+} \left( \phi_i + L_i + L_{LLQ} \right) \]

whereas for a standard queue it is:

\[ S_i^A = \frac{1}{C} \left[ F + N \cdot \phi_i - \phi_i + \sum_{j=1}^{N} L_j + L_{LLQ} \right] \]

Again, the start-up latency of the LLQ is reduced at the expenses of increasing the one of the standard queues. Note that the former does not depend on the number of active queues, even though it still accounts for one full quantum, i.e. for a possibly large number of packets. Figure 8 shows the ratio between the start-up latency of a queue in C-MDRR and in DRR, in the same settings as before.

5.2. J-MDRR

When a strictly-high priority queue is defined, its latency guarantees are the same as those of the LLQ in C-MDRR working in strict priority mode, i.e. are given by (15). Obviously, neither high nor low priority queues can have any latency guarantee in that case. Let us then assume that no strictly-high priority queue is defined. Let us denote with \( H \) (\( L \)) the subset of high- (low-) priority queues, and with \( F^H \) (\( F^L \)) the sum of their quanta, so that \( F = F^H + F^L \). The following result can be obtained by following the same proof process used for Theorem 1:

Theorem 2

"If no strictly-high priority queue is defined, the latency of a high-priority queue in J-MDRR is upper bounded by:

\[ \Theta_i^H = \frac{1}{C} \left[ (F - \phi_i) \left( 1 + \frac{L_i}{\phi_i} \right) + \sum_{j=1}^{N} L_j - F^L \right] \]

whereas the latency of a low priority queue is upper bounded by:

\[ \Theta_i^L = \frac{1}{C} \left[ (F - \phi_i) \left( 1 + \frac{L_i}{\phi_i} \right) + \sum_{j=1}^{N} L_j - F^L \right] \]

By comparing with (5), it can be straightforwardly shown that \( \Theta_i^H = \Theta_i^{DRR} - F/C \), and \( \Theta_i^L = \Theta_i^{DRR} \). So, the latency of a low priority queue under J-MDRR is equal to that of a DRR queue, and is not affected by the fact that a subset of queues is given higher priority. On the contrary, the latency of a high-priority queue in J-MDRR is reduced with respect to DRR. If only one high-priority queue is defined, (21) yields:

\[ \Theta_i^H = \frac{1}{C} \left[ \frac{L_i}{\phi_i} + \sum_{j=1}^{N} L_j \right] \]

Unlike the LLQ in alternate-priority C-MDRR (see (18)), the latency of a high-priority queue depends on its quantum, and can therefore be tuned by varying the latter. Figure 9 shows the ratio between the latency of a high-priority queue in J-MDRR and in DRR, in the same settings as before.

5.3. J-MDRR

When a strictly-high priority queue is defined, its latency guarantees are the same as those of the LLQ in C-MDRR working in strict priority mode, i.e. are given by (15). Obviously, neither high nor low priority queues can have any latency guarantee in that case. Let us then assume that no strictly-high priority queue is defined. Let us denote with \( H \) (\( L \)) the subset of high- (low-) priority queues, and with \( F^H \) (\( F^L \)) the sum of their quanta, so that \( F = F^H + F^L \). The following result can be obtained by following the same proof process used for Theorem 1:

Theorem 2

"If no strictly-high priority queue is defined, the latency of a high-priority queue in J-MDRR is upper bounded by:

\[ \Theta_i^H = \frac{1}{C} \left[ (F - \phi_i) \left( 1 + \frac{L_i}{\phi_i} \right) + \sum_{j=1}^{N} L_j - F^L \right] \]

whereas the latency of a low priority queue is upper bounded by:

\[ \Theta_i^L = \frac{1}{C} \left[ (F - \phi_i) \left( 1 + \frac{L_i}{\phi_i} \right) + \sum_{j=1}^{N} L_j - F^L \right] \]

By comparing with (5), it can be straightforwardly shown that \( \Theta_i^H = \Theta_i^{DRR} - F/C \), and \( \Theta_i^L = \Theta_i^{DRR} \). So, the latency of a low priority queue under J-MDRR is equal to that of a DRR queue, and is not affected by the fact that a subset of queues is given higher priority. On the contrary, the latency of a high-priority queue in J-MDRR is reduced with respect to DRR. If only one high-priority queue is defined, (21) yields:

\[ \Theta_i^H = \frac{1}{C} \left[ \frac{L_i}{\phi_i} + \sum_{j=1}^{N} L_j \right] \]

Unlike the LLQ in alternate-priority C-MDRR (see (18)), the latency of a high-priority queue depends on its quantum, and can therefore be tuned by varying the latter. Figure 9 shows the ratio between the latency of a high-priority queue in J-MDRR and in DRR, in the same settings as before.

6. SIMULATIONS

In this section we show some simulation results. We start with showing up to what extent J-MDRR prioritization is effective in
achieving average delay differentiation (whereas proofs in Section 5 were dealing with latency, and therefore with worst-case delays). Then, we compare C-MDRR and J-MDRR in a scenario in which sources transmit both real-time and non-real-time traffic. Simulations are carried out using ns-2 [22].

6.1. Average delay for J-MDRR
In the first experiment, we set up a J-MDRR scheduler managing a 10 Mbps link\(^1\), on which 10 queues are defined. Five queues are low priority queues, and five are high priority queues. All queues have a quantum equal to 1500 bytes, which implies that each queue is guaranteed a minimum bandwidth equal to 1 Mbps. In both subsets, 4 queues out of 5 are held always backlogged, while on the 5\(^{th}\) queue of each subset (the tagged queue) the offered load is varied from 10\% to 100\%. We plot the ratio of the average packet delay of the tagged high priority queue to the tagged low priority queue against their offered load in three different traffic scenarios:

- all queues transmit packets whose length is uniformly distributed between 100 and 1500 bytes.
- all queues transmit packets whose length is distributed according to a multimodal distribution, which is said to mimic the traffic observed in the Internet ([23], also shown in Table 1).

As Figure 10 shows\(^2\), the ratio is almost constant in both cases. In Figure 11 we show the average delay ratio when all the queues transmit 200 byte packets, with Poisson arrivals. In the experiment, the quantum is scaled upwards from 200 to 1000 bytes. The figure shows that the larger the quantum is (with respect to the packet length), the larger the average delay differentiation is. However, the delay differentiation is reduced as the offered load increases. When the queues are heavily loaded, both priority levels exhibit the same average delay.

<table>
<thead>
<tr>
<th>% pkts</th>
<th>Pkt length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.16</td>
<td>28</td>
</tr>
<tr>
<td>35.49</td>
<td>40</td>
</tr>
<tr>
<td>2.02</td>
<td>44</td>
</tr>
<tr>
<td>2.02</td>
<td>48</td>
</tr>
<tr>
<td>4.5</td>
<td>52</td>
</tr>
<tr>
<td>10.78</td>
<td>Uniform (40,80)</td>
</tr>
<tr>
<td>11.84</td>
<td>Uniform [80,576]</td>
</tr>
<tr>
<td>0.81</td>
<td>552</td>
</tr>
<tr>
<td>11.52</td>
<td>576</td>
</tr>
<tr>
<td>5.9</td>
<td>Uniform (576,1500)</td>
</tr>
<tr>
<td>0.96</td>
<td>628</td>
</tr>
<tr>
<td>2.99</td>
<td>1420</td>
</tr>
<tr>
<td>10.01</td>
<td>1500</td>
</tr>
</tbody>
</table>

Table 1 – multimodal packet length distribution

\(^1\) Although Cisco 1200 and Juniper M-series routers are able to manage Gbps-speed interface cards, we simulate low-bandwidth links in order to reduce the simulation time.

\(^2\) 95\% confidence intervals are too small to be visible, and therefore they are not shown.

6.2. Comparison in a single-node scenario
In this subsection we compare J-MDRR and C-MDRR in a single node scenario. We simulate a 100 Mbps link, on which voice, MPEG4 video, videoconference, and Internet-like data traffic are transmitted. Voice streams employ a G 711 codec, with voice activity detection. They transmit at 64Kbps during on periods, and the voice activity ratio is about 50\%. The MPEG4 streams are built from a video traffic generator based on TES (Transform Expand Sample) model of MPEG4 trace files [24], and they have an average bandwidth equal to 200Kbps each. Videoconference streams generate packets with a lognormal length distribution at 30 frames per second (with an average bandwidth of 180Kbps each). Internet-like data traffic consists in packets whose length is distributed according to the distribution reported in Table 1). Packet arrival times are Poisson-distributed, with an average configured so as to achieve a desired rate.

We set up one queue per traffic class, and give voice traffic the highest priority. All queues have 2Mbytes of reserved buffer space, and quanta, weights and priorities are provisioned as shown in Table 2. Note that the weights and quanta are selected so as to achieve approximately the same bandwidth sharing ratio in the two algorithms (for C-MDRR, we assume an MTU of 1500 bytes). The number of sources hashing on the same queue is shown in Table 3. Note that the quantum (weight) selection is such that the MPEG and the videoconference queue are given 10\% of the bandwidth left unused by voice traffic (i.e., about 9.7 Mbps), while data traffic is given the remaining 80\%, (i.e. 77.4 Mbps). The MPEG and videoconference queues are therefore overprovisioned of about 40\% with respect to their average offered load.
### Table 2 – scheduler settings

<table>
<thead>
<tr>
<th>Traffic class</th>
<th>Weight</th>
<th>Pri.</th>
<th>Quantum</th>
<th>Pri.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoIP</td>
<td>LLQ (strict)</td>
<td>-</td>
<td>-</td>
<td>Str.high</td>
</tr>
<tr>
<td>MPEG</td>
<td>1</td>
<td>Std.</td>
<td>1500</td>
<td>High</td>
</tr>
<tr>
<td>VCF</td>
<td>1</td>
<td>Std.</td>
<td>1500</td>
<td>High</td>
</tr>
<tr>
<td>Data</td>
<td>10</td>
<td>Std.</td>
<td>12000</td>
<td>Low</td>
</tr>
</tbody>
</table>

### Table 3 – no. of sources and bandwidth for real time traffic classes

<table>
<thead>
<tr>
<th>Traffic class</th>
<th># of sources</th>
<th>Av. offered load</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoIP</td>
<td>100</td>
<td>3.2 Mbps</td>
</tr>
<tr>
<td>MPEG</td>
<td>35</td>
<td>7 Mbps</td>
</tr>
<tr>
<td>VCF</td>
<td>39</td>
<td>7 Mbps</td>
</tr>
</tbody>
</table>

In Figure 12-15 we show the average delay of a VoIP, MPEG4 and videoconference source as a function of the offered load on the data queue. We also plot the 95% confidence intervals, computed on 30 independent simulation runs for each point in the graph. The delay for the VoIP queue, which is serviced at the highest priority, is exactly the same in both algorithms. However, as the offered load on the data queue increases, MPEG and video-conference streams exhibit a smaller average delay in J-MDRR (by 30 to 50% when the offered load is equal to 100%). In fact, in C-MDRR, a packet of an MPEG stream arriving at an empty queue may have to wait for a whole quantum to be serviced from both the data and the videoconference queue. On the contrary, in J-MDRR it would only have to wait for a packet from both queues.

### 7. CONCLUSIONS

In this paper we have analyzed the performance of two variants of the Modified Deficit Round Robin scheduler, implemented on Cisco 12000 and Juniper M-Series routers. The analysis has been carried out by deriving the minimum transmission rate and the latency for each queue under the different priority regimes, and by simulation. The analysis shows that, in Cisco MDRR:

- when alternate priority is activated, it may not be possible that a set of flows requesting less than 100% of the channel bandwidth be actually guaranteed the required rate;
- the mechanisms for quantum selection allow a limited control over bandwidth sharing. More specifically, it is not possible to allocate bandwidth in a purely weighted way;
- the latency of the LLQ can only be reduced at the expense of increasing it for the standard queues. Furthermore, the latency of all queues cannot be effectively controlled by varying the queue weight.

On the other hand, in Juniper MDRR, a subset of queues can be prioritized so as to have smaller latency, without increasing the latency of the other queues. Furthermore, simulations show that such queue prioritization is effective also for achieving differentiation among average delays.

### 8. ACKNOWLEDGEMENTS

This work was supported by the Italian Ministry of Education, Universities and Research (MIUR) in the framework of the FIRB-Perf project.

### 9. REFERENCES


APPENDIX

Proof of Theorem 1

Hereafter, we only report the proof of (16) due to space constraints. The same process can also be employed in order to prove (17). Let us denote with $W(t_i,t)$ the service received by a queue $i$ in the time interval $[t_i,t)$ (service curve of flow $i$). We assume that $W(t_i,t)$ increases by a quantity equal to the packet length when the last bit of the packet has been serviced. In order to derive the latency, we apply the methodology described in [2], Lemma 7. Specifically, we assume that the LLQ is continuously backlogged starting from time $t_i$ and we prove that, in a generic time interval $[t_i,t)$, it is:

$$W(t_i,t) \geq \left[ R_{LLQ}^d (t - t_i - \Theta_{LLQ}) \right]$$

The latter representing the latency rate curve of the LLQ. As (24) is required to hold in any possible scenario, we can limit ourselves to proving it when $W(t_i,t)$ is the lowest possible. Clearly, this happens in a scenario (hereafter worst-case scenario) in which:

a. The LLQ has each service opportunity as late as possible;

b. the service received by the LLQ after each service opportunity is the lowest possible.

Without loss of generality, we will assume $t_i=0$. We denote with $T_i$ the time instant in which the $v^{th}$ service opportunity for the LLQ ends.

Given the way quotas are allocated in C-MDRR, we can limit to considering $\phi_{LLQ} \geq L_{LLQ}$, and $\phi \geq T_i$ for all standard queues, in which case a backlogged flow can transmit at least one packet on each service opportunity. For mathematical convenience, we need to define two functions which bound the service curve $W(0,t)$:

- $W(0,t)$, which increases with a slope $C$ whenever the LLQ is being serviced. More formally:
  - for any instant $p$ in a time interval $[p_i,p_2) \subseteq [0,t]$ in which the LLQ is not being serviced, $W(0,p) = \bar{W}(0,p_1)$, i.e. $\bar{W}(0,p)$ remains constant;
  - for any instant $p$ in a time interval $[p_i,p_2) \subseteq [0,t]$ in which the LLQ is being serviced, $\bar{W}(0,p) = \bar{W}(0,p_1) + C \cdot (p - p_1)$, i.e. $\bar{W}(0,p)$ increases with a slope $C$;
  - $w(0,t)$, which increases stepwise with a step equal to the received service at the end of a time interval in which the LLQ is being serviced and remains constant elsewhere.

Clearly, for any time instant $t$, the following relationship holds:

- $w(0,t) \leq W(0,t) \leq \bar{W}(0,t)$ when the LLQ is being served
- $w(0,t) = \bar{W}(0,t) = \bar{W}(0,t)$ when the LLQ is not being served.

Figure 15 shows the three curves. For the sake of readability, the channel capacity is assumed to be equal to 1 in the figures, so that
both the horizontal and vertical axis have the same scale.

\[
W(0,t) = v \cdot \phi_{LLQ} - \frac{L_{LLQ}}{C} \tag{25}
\]

This implies that \( W(0,t) \) increases by a step equal to \( \phi_{LLQ} \) at each time instant \( T_s, \ v > 1 \). When \( v = 1, \ w(0,t) \) increases by a step equal to \( \phi_{LLQ} - \frac{L_{LLQ}}{C} \). However, in a finite capacity system, the service curve cannot increase by a step greater than the maximum length packet; therefore we can then obtain a tighter lower bound on the service curve than \( w(0,t) \) by considering:

\[
w(0,t) = \max \{ w(0,t), W(0,t) - \frac{L_{LLQ}}{C} \}
\]

The curve \( w'(0,t) \) is shown in Figure 16.

\[
\begin{align*}
\mathcal{W}(0,t) & = v \cdot \phi_{LLQ} - \frac{L_{LLQ}}{C} \\
W(0,t) & = v \cdot \phi_{LLQ} - \frac{L_{LLQ}}{C} \\
w(0,t) & = \max \{ w(0,t), W(0,t) - \frac{L_{LLQ}}{C} \}
\end{align*}
\]

**Figure 16 - Improved bounds on the service curve**

Define \( X = \left\{ j : 1 \leq j \leq N, \phi_j + L_j \geq \phi_{\text{max}} \right\} \), the set of standard queues whose maximum deficit can actually exceed the maximum length quantum. That set includes the standard queue with the maximum quantum itself. Assume for simplicity of notation that the standard queues in the above set are sorted by decreasing maximum quantum. That set includes the standard queue with the maximum quantum itself. Assume for simplicity of notation that the standard queues in the above set are sorted by decreasing maximum quantum. The above-mentioned line is the latency-rate curve for the LLQ, and \( \Theta_{LLQ} \) is the LLQ latency. In fact, it is straightforward to show that the curve:

\[
s(t) = \left[ C \cdot \frac{\phi_{LLQ}}{\phi_{\text{max}} + \phi_{LLQ}} \cdot t - \Theta_{LLQ} \right]^{-1}
\]

i) has at least one intersection with \( w'(0,t) \); ii) is such that \( w'(0,t) \geq s(t), t \geq 0 \).

This completes the proof.

- Each queue \( j \in X \) starts with the maximum deficit \( \Delta_j = L_j \).
- Each queue \( j \in X \) always transmits packets in such a way that its quantum is always fully exploited (i.e., no deficit is carried over onto subsequent rounds).
- The active list contains standard queues sorted by decreasing value of \( \phi_j + L_j \) at time \( t = 0 \).

Under the above conditions, we have:

\[
T_k = \begin{cases}
\frac{1}{C} (\phi_{LLQ} - \frac{L_{LLQ}}{C} + \phi_k + L_k) & k = 1 \\
T_{k-1} + \frac{1}{C} (\phi_{LLQ} + \phi_k + L_k) & 1 < k \leq |X| \\
T_{k-1} + \frac{1}{C} (\phi_{LLQ} + \phi_{\text{max}}) & k > |X|
\end{cases}
\]

More specifically, when \( k = |X| \) we have:

\[
T_{P1} = \frac{1}{C} \left[ \frac{|X| \phi_{LLQ} - \frac{L_{LLQ}}{C} + \sum_{j=1}^{\phi_{\text{max}}} (\phi_j + L_j)}{\sum_{j=1}^{\phi_{\text{max}}} (\phi_j + L_j)} \right]
\]

Let \( s_j \) denote the time instant that lies \( \left( \frac{\phi_{LLQ} - \frac{L_{LLQ}}{C}}{C} \right) \) units of time before \( T_c \) (note that \( T_c = s_j \) if \( \phi_{LLQ} = L_{LLQ} \)). We observe that, for \( k \geq |X| \),

\[
\frac{w(0,s_{j+1}) - w(0,s_j)}{s_{j+1} - s_j} = C \cdot \frac{\phi_{LLQ}}{\phi_{\text{max}} + \phi_{LLQ}} \frac{L_{LLQ}}{C}
\]

which confirms that the worst-case rate for the LLQ is the one shown in Section 4.1.2. Consider the line that joins points \( (s_j, w(0,s_j)), \ v \geq |X| \), and denote with \( \Theta_{LLQ} \) its intersection with the time axis. After some straightforward algebraic manipulations, we obtain:

\[
\Theta_{LLQ} = \frac{1}{C} \left[ \sum_{j=1}^{\phi_{\text{max}}} (\phi_j + L_j) - \frac{|X|}{|X|} (\phi_{\text{max}} + L_{LLQ}) \right]
\]

\[
\Theta_{LLQ} = \frac{1}{C} \left[ \sum_{j=1}^{\phi_{\text{max}}} (\phi_j + L_j) - \frac{|X|}{|X|} (\phi_{\text{max}} + L_{LLQ}) \right]
\]

\[
\Theta_{LLQ} = \frac{1}{C} \left[ \sum_{j=1}^{\phi_{\text{max}}} (\phi_j + L_j) - \frac{|X|}{|X|} (\phi_{\text{max}} + L_{LLQ}) \right]
\]

\[
\Theta_{LLQ} = \frac{1}{C} \left[ \sum_{j=1}^{\phi_{\text{max}}} (\phi_j + L_j) - \frac{|X|}{|X|} (\phi_{\text{max}} + L_{LLQ}) \right]
\]

We observe that it is not possible for the deficit variable to assume the value of the maximum length packet, since inequality holds into (1).