Performance Tuning of Dual-priority Delta Networks through Queuing Scheduling Disciplines

Dimitris C. Vasiliadis, George E. Rizos, and Costas Vassilaki

Original scientific paper

Asbstract—Differentiated Services (DiffServ) and other scheduling strategies are now widespread in the traditional, "best effort" Internet. These Internet Architectures offer Quality of Service (QoS) guarantees for important customers at the same time as supporting less critical applications of lower priority. Strict priority queuing (PQ), weighted round robin (WRR), and class-based weighted fair queuing (CBWFQ) are three common scheduling disciplines for differentiation of services in telecommunication networks. In this paper, a comparative performance study of the above PQ, WRR and CBWFQ queuing scheduling policies applied on a double-buffered, 6-stage Multistage Interconnection Network (MIN) that natively supports a 2-class priority mechanism is presented and analyzed using simulation experiments. We also consider a 10-stage MIN, to validate that the conclusions drawn from the 6-stage MIN apply to MINs of different sizes. The findings of this paper can be used by MIN designers to optimally configure their networks.

***Index Terms*—Diffserv networks, Multiple access scheduling algorithms, Multistage Interconnection Networks, Performance Evaluation, Simulation**

I. INTRODUCTION

RECENTLY,, there has been much research conducted on the development of Quality of Service (QoS) on IP networks, keeping in mind the goal of allowing network operators to offer diverse levels of treatment to packets traversing their networks. The QoS of today's best-effort network does not deliver the performance required for a wide range of interactive and multimedia applications that have demanding delay and throughput requirements. On the other hand, the two predominant networking architectures that consider the problem of providing QoS for a given IP packet in the internet are Integrated Services (IntServ), documented in RFC 1633, RFC 2212, and RFC 2215 [23] and Differentiated Services (DiffServ), defined in RFC 2474 and RFC 2475 [22]. Integrated Services came first and were based on building a virtual circuit through the internet using the bandwidth reservation technique. IntServ exhibits however scalability problems, being not recommended for core networks [9], since the task of reserving paths would be very tedious in a busy network such as the Internet. DiffServ then tried to answer some of the problems that came up using IntServ by differentiating the traffic. Integrated Services, which is also called "Hard QoS", can reserve resources or bandwidth between networking devices by employing protocols like RSVP [46] (Resource Reservation Protocol), while Differentiated Services, which is also called "Soft QoS" is using a tag in the Network Layer to mark the packets' priority. Subsequently, a number of packet scheduling algorithms have been proposed, with the most prominent ones including strict priority queuing (SPQ) [20] [also known as simply "priority queuing" (PQ)], round-robin (RR) [44] and its variations – e.g. weighted round-robin (WRR) [4], deficit round-robin (DRR) [27], and smoothed round-robin (SRR) [10]) –, generalized processor sharing (GPS) [6], weighted fair queuing (WFQ)[18] [also known as packet-by-packet generalized processor sharing (P-GPS)] [5], class-based weighted fair queuing (CBWFQ) [26][45], and a hybrid algorithm named PQ-CBWFQ called also low latency queuing (LLQ) [12].

Regarding the network switch internal architecture, Multistage Interconnection Networks (MINs) with crossbar switching elements (SEs) are often used as communications infrastructure in the domains of networked systems and multiprocessor systems [31] and have also recently been identified as an efficient communication backplane of high-performance networking elements including gigabit Ethernet switches, terabit routers and ATM switches [28], [32], while they are also used as underpinnings for delivering broadband services [15].The spread of MINs can be attributed to their potential to concurrently route multiple packets resulting in good exploitation of the available hardware and the ability to provide increased fault tolerance [17] as well as to their appealing cost/performance ratio, which is quite small, compared to other approaches.

In the current literature, the performance of multi-priority MINs under the strict priority queuing algorithm has been studied extensively through both analytical methods and simulation experiments (e.g. [36], [37], [33]), considering various buffer sizes (mainly buffers of sizes 1, 2 and 4), buffer size allocation to different priority classes (symmetric vs. asymmetric [37]), arrival processes (e.g. uniform vs. bursty [11]), traffic patterns (e.g. uniform vs. hotspot [39],[40];

Manuscript received September 2, 2013; revised January 21, 2016.

Dimitris C. Vasiliadis is with the University of Peloponnese, Department of Informatics and Telecommunications, TermaKaraiskaki, 22100, Tripoli, Greece and with the Network Operations Center, Technological Educational Institute of Epirus, Kostakioi, 47100 Arta, Greece (e-mail: dvas@uop.gr).

George E. Rizos is with the University of Peloponnese, Department of Informatics and Telecommunications, TermaKaraiskaki, 22100, Tripoli, Greece and with the Network Operations Center, Technological Educational Institute of Epirus, Kostakioi, 47100 Arta, Greece (e-mail: georizos@uop.gr).

Costas Vassilakis is with the University of Peloponnese, Department of Informatics and Telecommunications, TermaKaraiskaki, 22100, Tripoli, Greece (e-mail: costas@uop.gr).

unicast vs. multicast [13],[29]) and internal MIN architectures (e.g. single-layer vs. multi-layer [34]). These studies have shown that under increased network load (packet arrival probability $\lambda > 0.6$) the QoS offered to low priority packets rapidly deteriorates, with throughput significantly dropping and delay sharply increasing. While strict priority queuing is the "gold standard" for high priority traffic, weighted round-robin and class-based weighted fair queuing scheduling disciplines appear as a plausible solution for providing better QoS to low-priority packets under increased network load, since one of the goals of these scheduling techniques is to increase fairness, giving low-priority queues the opportunity to transmit packets even though the high-priority queues are not empty. Insofar, however, there are no studies to quantify and compare the gains obtained for low-priority packets (and conversely the losses incurred for high-priority packets) by employing the above queuing packet scheduling algorithms on Multistage Interconnection Networks.

In this paper, we investigate how the performance of the three dominant scheduling algorithms used in MIN networks, namely PQ, WRR and CBWFQ, can be tuned through the individual algorithms' parameters. This investigation is conducted by means of a simulation-based comparative performance study, targeting dual-priority, double-buffered, 6-stage MINs. In this performance study, we calculate and compare the QoS offered to packets of different priority classes, under varying loads, specific ratios of high/low priority packets and different priority-weights of service within the overall network traffic. The high priority to low priority packet ratios have been chosen to correspond both to the recent trends in network traffic, where traffic with real-time requirement dominates (e.g. [3] predicts that two-thirds of the world's mobile data traffic will be video by 2017) as well as corporate networking environments where business-oriented applications do not typically exhibit real-time characteristics. The findings of this paper can be used by MIN designers in predicting the performance of their networks before actual network implementation, and in understanding the impact of the examined scheduling algorithms and their parameters on the QoS offered to packets of different priorities.

Our work extends the results of the published literature as follows:

1. it examines and compares the behavior of all three dominant scheduling algorithms offered by network devices.
2. it considers a *normal scenario*, corresponding to the current analogy of high/low priority packets) and an *extreme scenario*, corresponding to the projected analogy of high/low priority packets in the near future. We believe that this insight is valuable for network designers and administrators, allowing them to prepare their networks for the forthcoming traffic patterns.
3. we examine networks of different sizes, allowing us to generalize our conclusions regarding the behavior of these algorithms.

The rest of this paper is organized as follows: in section II we briefly present the commonly used scheduling algorithms considered in this paper, while in section III we analyze a 2-class priority MIN and give details on its operation under different queuing scheduling disciplines. In sections IV and V we present the performance metrics and the simulation results, respectively, while in section VI conclusions are drawn and future work is outlined.

II. Analysis of Queuing Scheduling Algorithms

Among the commonly used queuing algorithms, strict priority queuing (PQ) [20] [4] [12] scheduling allows higher priority packets to be de-queued and sent before any other packet in lower-priority queues. Strict priority queuing assumes that types of traffic can be differentiated and treated preferentially. Priority queuing can support multi-priority classes, usually ranging from high to low. Separate FIFO queues are created for each defined priority level and the arriving traffic is sorted into its proper queue as it arrives. Queues are serviced in strict order of priority, so the high priority queue always is serviced first; only if the high priority queue is empty the next-lower priority is served, and so on. Consequently, if a lower-priority queue is not empty and a packet enters a higher queue, the processing of the lower-priority queue would be deferred in favor of the higher-priority queue. This mechanism is good for prioritizing delay-sensitive high-priority data such as voice, but unfortunately this policy may lead to the low priority traffic to starvation, especially when the offered load of higher priority packets is considerable. In fact, strict priority queuing is the gold standard for high priority traffic, giving high priority traffic the best treatment among all available policies. On the other hand, if the amount of high-priority traffic is great, other will rarely get the chance to transmit packets, leading to considerably worse performance for lower-priority packets than if a single FIFO queue were used.

Round-robin (RR) [43] [44] is one of the simplest scheduling algorithms. It was initially used for processes scheduling in operating systems, assigning time slices to each process in equal portions and in circular order, handling all processes without priority (also known as *cyclic executive*). Round-robin scheduling is both simple and easy to implement, being also starvation-free. Round-robin scheduling is also applied to other scheduling problems, including data packet scheduling in computer networks. To augment RR with the capability to offer diverse treatment to data packets, the weighted round-robin (WWR) scheduling method [4] was introduced, where the weighting factor applied to a specific class queue determines how many bytes of data the processor delivers from this queue before it moves on to the next one. Thus, as WRR mechanism cycles through the queues, servicing an amount of packets from each queue until the number of bytes transmitted exceeds the bandwidth determined by the queue's weighting factor or the queue is empty. In the case of an empty queue, it is obvious that the corresponding switching element will try to forward packets from the next active queue that has packets ready to send.

WRR algorithm uses a predefined relative weight for each priority-class queue defining thus the percentage of time the processor services the specific queue before moving on to the next one. This mechanism prevents the head-of-line blocking that can occur with a strict priority queuing (PQ) scheduling.

Weighted fair queuing (WFQ) [12] is a data packet scheduling technique allowing different sessions to have different service shares. In WFQ, with a link data rate of *R*, if *N* data flows currently are active, with weights $w_1,w_2...w_N$, data flow number *i* will achieve an average data rate of $R*w_i/(w_1+w_2+...+w_N)$ [26]. Native weighted fair queuing (WFQ) assigns a weight to each conversation, and then schedules the transmit time for each packet of the different flows. The weight is a function of the IP precedence of each flow, and the scheduling time depends usually on the packet size. WFQ was implemented for slow speed links (such as serial) to provide a fair treatment for each type of traffic. On the other hand, class-based weighted fair queuing (CBWFQ) [26] [38] is usually applied at high speed networks (such as ATM) providing user defined traffic classes and allowing more control and functionality than weighted fair queuing (WFQ) scheduling. CBWFQ uses matching criteria obtained i.e. by Network Based Application Recognition (NBAR), or Access Control Lists (ACLs). A queue is reserved for each class and matching traffic is directed to that queue. CBWFQ divides user traffic into a hierarchy of classes based on criteria which may consider any combination of IP addresses, protocols and application types. Since a company's accounting department, for example, may not need the same Internet access privileges as the engineering department, it is vital for traffic management technology to provide flexibility and granularity in classifying traffic flows.

The selection of the packet scheduling algorithm can drastically affect the quality of service observed by the packets traversing the network and the overall network performance, since different algorithms aim to optimize different metrics of packet QoS, such as delay, delay jitter, throughput and fairness, while additional factors –e.g. multiuser diversity [14]- can be also considered. Other algorithm properties that are taken into account for choosing the packet scheduling algorithm that will be implemented in a network are its space and time complexity [10] (since they affect the memory and the processing required to implement the algorithm, respectively) and the ease of implementation, since more complex algorithms are generally more demanding in space and time and their implementations are more prone to errors. The packet scheduling algorithms outlined above have been adopted by the industry and implemented in most commercial products (e.g. [19], [2], [4], [12]) mainly due to their following characteristics (a) they are easy to implement and verify (b) they exploit well the available network bandwidth (c) they have very small memory processing power requirements and (d) network administrators find them easy to understand and configure.

III. Two-Class Priority MIN and Queuing Scheduling Disciplines

Multistage Interconnection Networks (MINs) are used to interconnect a group of *N* inputs to a group of *M* outputs using several stages of small size Switching Elements (SEs) followed (or preceded) by link states. All types of blocking Multistage Interconnection Networks (Delta Networks, Omega Networks and Generalized Cube Networks) with the Banyan property [8] are self-routing switching fabrics and they are characterized by the fact that there is exactly a unique path from each source (input) to each sink (output).

A typical configuration of a Multistage Interconnection Network is depicted in Fig 1. Each (2X2) SE has been implemented using two transmission queues per link [24], [25], accommodated in two (logical) buffers, with one queue dedicated to high priority packets and the other dedicated to low priority ones. Under this scheme, the queue scheduling algorithm selects the next packet to be forwarded from each SE's queues to the outgoing links of the SE. In this paper, we consider three different queuing scheduling algorithms (PQ, WRR and CBWFQ) which are applied on a banyan-type dual-priority Multistage Interconnection Network operating under the following assumptions:

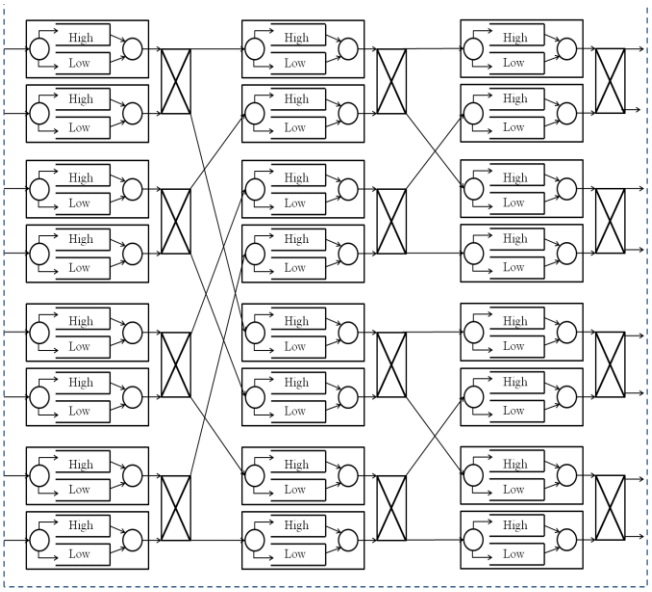

Fig. 1.An (8X8) Dual-priority MIN.

- Routing is performed by all SEs in parallel, thus the MIN can be considered to operate in a pipeline fashion. The pipeline is synchronized using an internal clock and operates in a slotted time model [30]. The service time for all SEs is deterministic.
- Each input of the MIN accepts only one packet within each time slot. A packet entering the MIN comprises of (i) the routing tag, which effectively contains the routing instructions for all SEs that the packet will traverse (ii) the packet priority; since in this paper we consider a dual-priority scheme, the priority specification is a single bit designating the packet as high- or low-priority one and (iii) the packet payload, i.e. the actual data that are sent to the

destination. Upon reception, packets are first classified according to their priority, and are then assigned to the queue specifically dedicated to the particular priority (Fig. 2). Both high- and low-priority packets are considered to be uniformly distributed across all destinations.

- All packets have the same size, arrivals are independent of each other and packets arrive with equal probability at all inputs. The arrival process of each input of the network is a simple Bernoulli process. We will denote this probability as λ. This probability can be further broken down to λ_h and λ_l, which represent the arrival probability for high and low priority packets, respectively. It holds that $\lambda = \lambda_h+\lambda_l$.
- All SEs operate in a store-and-forward fashion, i.e. each packet received by a SE is stored in a buffer until it can be forwarded to the next SE (or sent to the MIN output, if the SE is at the last stage of the MIN). To enable its store-and-forward operation, each SE incoming link is associated with two FIFO buffers, one dedicated to high-priority packets and one dedicated to low-priority ones. When a FIFO buffer of a specific (high or low) priority-class queue within a SE is full, the SE cannot accept further input packets of the same priority-class from its predecessor SEs (or the MIN input), and a backpressure mechanism is employed to force this priority-class packets to remain in the previous MIN stage until amble buffer space is available. Under this scheme, no packets are lost inside the MIN.

  Regarding the PQ scheduling algorithm the low-priority queue is only serviced if the high-priority queue contains no packets. On the other hand, WRR uses a weight value to decide how many packets to transmit from one queue before it switches to the other queue. The higher the weight assigned to a queue, the more transmission bandwidth is allocated to it. For example in a dual-priority system with w_h=7 and w_l=3 the weighted round-robin algorithm checks first, for 7 continuous time-slots, the high-priority queue of each SE, while for the next 3 time-slots the low-priority queue of each SE is examined first for forwarding a ready packet. Similarly, CBWFQ assigns a weight to each class, which determines the transmit order for queued packets. Each packet priority queue is statically assigned a weight, which specifies the bandwidth ratio that will be dedicated to the particular queue. Naturally, the sum of all weights must be equal to 1. At each network cycle, the class-based weighted fair queuing algorithm examines the priority queues to select the packet to be forwarded through the output link, always observing the bandwidth ratio that has been assigned to each queue. A prominent method for achieving this is to determine the set S of non-empty queues in the system and choosing a queue among them with probability $p(q_i) = w_i / \sum_{j \in S} w_j$, where w_i is the weight assigned to i-queue [26]. This is analogous to lottery scheduling used in operating systems [42].
- All queuing scheduling algorithms considered in this paper are *work conserving*, i.e. a packet is always transmitted when there is traffic waiting, as opposed to *non-work conserving* algorithms which do not transmit a packet if the queue whose turn is to transmit a packet is found to be empty [16]. If a queue does not use its bandwidth ratio within a time window, this bandwidth is divided among the queues that do have packets to transmit, proportionally to their weights.
- The contention is solved randomly with equal probabilities. Thus, when two packets at a stage contend for a buffer at the next stage and there is no adequate free space for both of them to be stored (i.e. only one buffer position is available at the next stage), one packet will be accepted at random and the other will be blocked by means of upstream control signals. Note that since packets of different priorities are stored in different queues, the contention for buffer space always occurs between packets of the same priority.
- Packets are removed from their destinations immediately upon arrival, thus packets cannot be blocked at the last stage.

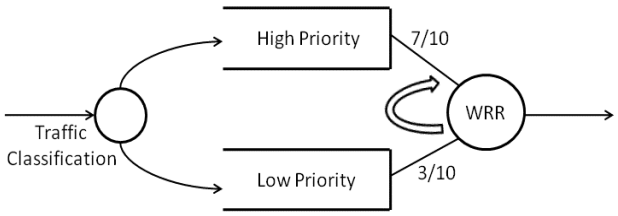

Fig. 2. Weighted Round-Robin Queuing Algorithm.

IV. Performance Evaluation Operational Parameters and Metrics

A. MIN Configuration and Traffic Load Parameters

In this paper we extend previous studies on performance evaluation of MINs by comparing the performance metrics of a 2-class priority MIN under various queuing scheduling disciplines and under different traffic load scenarios. More analytically, we consider and compare the performance of the three aforementioned scheduling algorithms (PQ, WRR and CBWFQ), while the operational parameters of the MIN evaluated in this chapter are as follows:

- *Buffer size* (b) of a queue is the maximum number of packets that an input buffer of a SE can hold. In this study, symmetric double-buffered SEs (b=2) are considered for both high- and low-priority packets of MIN, since blockings can occur and thus additional buffers may be needed to store blocked packets and newly arriving packets. We note here that the particular *buffer size* has been chosen since it has been reported [35] to provide optimal overall network performance, offering high throughput and avoiding excessive increases of delay.
- *Number of stages n* is the number of stages of an (N X N) MIN, where n=$\log_2 N$. In our case study n is assumed to be6, thus the MIN size is (64 X 64).
- *Offered load (λ)* is the steady-state fixed probability of arriving packets at each queue on inputs. In our simulation λ is assumed to be λ = 0.1, 0.2… 0.9, 1.λ can be further

broken down to λ_h and λ_l, which represent the arrival probability of the high and low priority traffic of the offered load respectively. It holds that $\lambda=\lambda_h+\lambda_l$.

- *Ratio of high priority packets* (r_h), is the ratio of high priority offered load, where $r_h = \lambda_h/\lambda$; this traffic is uniformly distributed among all output ports. In our study r_h is assumed to be r_h =0.30 (normal scenario) or 0.70 (extreme scenario). It is obvious that the corresponding ratio *of low priority packets* (r_l), where $r_l = \lambda_l/\lambda$, is always equal to 1 -r_h and is therefore assumed to be r_l =0.70 or 0.30 respectively.
- *Weight of high priority queues* (w_h), is the percentage rate of processor dedicated to high priority packets in each queue by the applied scheduling algorithm. In WRR discipline the w_h factor determines the number of continues cycles in which the high-priority queues are examined first and subsequently serviced if the corresponding queues are not empty. It is obvious that the *weight of low priority* queues(w_l) declares in the same way the number of cycles in which the low-priority queues are examined first. Considering the above two scenarios for *high priority ratios* (30% or 70%), w_h is assumed to be6 or 8, whereas the corresponding w_l is configured to be 4 or 2 respectively. Naturally, in both cases the ratio $w_h/(w_h+w_l)$ is configured to a higher value than the corresponding r_h, so as to offer better QoS to high priority packets. Similarly, in the case of CBWFQ discipline the corresponding w_h is assumed to be0.6 or 0.8, whereas the w_l is considered to be 0.4 or 0.2 respectively, expressing thus the probability that a particular class queue is examined first; this probabilistic mechanism applied individually in each SE for every cycle repeatedly.

B. MIN Performance Metrics

In this section the two most important network performance factors, namely *packet throughput* and *delay* are evaluated and analyzed under the operational parameters of MIN described previously. We also investigate the *universal performance factor* introduced in [35] and also used in [41], which combines the above two metrics into a single one. The following metrics are used in order to evaluate the performance of a (*NXN*) MIN.

Let *Th* and *D* be the normalized throughput and normalized delay of a MIN.

- *Relative normalized throughput RTh*(*h*) of high priority packets is the *normalized throughput Th*(*h*) of such packets divided by the corresponding ratio of *offered load* r_h.

$$RTh(h) = \frac{Th(h)}{r_h} \tag{1}$$

- Similarly, *relative normalized throughput RTh*(*l*)of low priority packets can be expressed by the *ratio of normalized through put Th*(*l*) of such packets to the corresponding *ratio of offered load* r_l.

$$RTh(l) = \frac{Th(l)}{r_l} \tag{2}$$

This extra normalization of both high and low priority traffic leads to a common value domain needed for comparing their absolute performance values in all configuration setups.

- *Universal performance factor Upf* is defined by a relation involving the two major above normalized factors, *D* and *Th*[35]: the performance of a MIN is considered optimal when *D* is minimized and *Th* is maximized, thus the formula for computing the *universal factor* arranges so that the overall performance metric follows that rule. Formally, *Upf* can be expressed by

$$Upf = \sqrt{w_d * D^2 + w_{th} * \frac{1}{Th^2}} \tag{3}$$

Where w_d and w_{th} denote the corresponding *weights* for each factor participating in the *Upf*, designating thus its importance for the corporate environment. Consequently, the performance of a MIN can be expressed in a single metric that is tailored to the needs that a specific MIN setup will serve. It is obvious that, when the *packet delay* factor becomes smaller or/and *throughput* factor becomes larger the *Upf* becomes smaller, thus smaller *Upf* values indicate better overall MIN performance. Because the above factors (parameters) have different measurement units and scaling, they are normalized them to obtain a reference value domain. Normalization is performed by dividing the value of each factor by the (algebraic) minimum or maximum value that this factor may attain. Thus, equation (3) can be replaced by:

$$Upf = \sqrt{w_d * \left(\frac{D - D^{\min}}{D^{\min}}\right)^2 + w_{th} * \left(\frac{Th^{\max} - Th}{Th}\right)^2} \tag{4}$$

Where D^{min} is the minimum value of *normalized packet delay(D)* and Th^{max} is the maximum value of *normalized throughput*. Consistently to equation (3), when the *universal performance factor Upf*, as computed by equation (4) is close to 0, the performance a MIN is considered optimal whereas, when the value of *Upf* increases, its performance deteriorates. Moreover, taking into account that the values of both *delay* and *throughput* appearing in equation (4) are normalized, $D^{\min} = Th^{\max} = 1$, thus the equation can be simplified to:

$$Upf = \sqrt{w_d * (D-1)^2 + w_{th} * \left(\frac{1-Th}{Th}\right)^2} \tag{5}$$

The extra normalization of both high and low priority traffic considered in the evaluation of *relative normalized throughput* leads to the following formula at dual-priority MINs

$$Upf(p) = \sqrt{w_d * (D(p)-1)^2 + w_{th} * \left(\frac{1-RTh(p)}{RTh(p)}\right)^2} \tag{6}$$

Where $p=\{h, l\}$ stands for high and low priority traffic respectively. In this study, when calculating the value of this combined factor, we have considered the individual performance factors (*packet throughput* and *delay*) to be of

equal importance, setting thus w_d=w_{th}=1. However, for some specific application classes, e.g. batch data transfers (where the *throughput* is more important) or streaming media (where the *delay* must be optimized), different weight values for the above metrics would be considered.

V. Simulation And Performance Results

Recall from section III that in this paper we assume that all packets entering the MIN have the same size; this behavior is similar e.g. to what would be expectedly found in fix-length, cell-based ATM networks. Under fixed-length packet traffic, both WRR and CBWFQ scheduling disciplines are proved to be fair in serving process [27]. In the presence however of variable-length packet sizes, the service class with the larger average packet size obtains more than its configured share of output port bandwidth. In that case, the DWRR [27] queuing algorithm would be similarly applied to guarantee fair queue serving.

In order to perform the experiments presented in this paper, we developed a special simulator in C++, capable of handling finite-buffered MINs natively supporting 2-class priority. The simulator encompasses implementations of the three queue scheduling algorithms considered in this paper, namely PQ, WRR and CBWFQ.

Packets entering the MIN are appropriately tagged as either high- or low-priority. Internally, each (2X2) SE was modeled using four non-shared buffer queues, where buffer queue operation was based on the first come first serviced (FCFS) principle; the first two buffer queues dedicated to high priority packets (one per incoming link), and the other two dedicated to low priority ones. When an SE receives a packet, it first classifies it i.e. it examines its priority tag and then inserts it in the appropriate queue. Subsequently queues are served according to the scheduling discipline in effect for the particular simulation run.

In this simulator, several operational parameters have been modeled and can be defined by the user, such as the *buffer-length,* the *number of input and output ports*, the *ratio of high-priority packets,* the *offered load*; when either WRRR or CBWFQ are employed, the *weight of high-priority queues* can be also defined.

Finally, the simulations were performed at packet level, assuming fixed-length packets transmitted in equal-length time slots, while the number of simulation runs was again adjusted at 10^5 clock cycles with an initial stabilization process 10^3 network cycles, ensuring a steady-state operating condition.

A. Simulator Validation

Our simulator was validated by comparing the results obtained from our simulation experiments against those reported in other works that also examine MINs natively supporting a dual-priority mechanism, and selecting among them the ones considered most accurate under uniform traffic conditions. Since no other related works on simulators for dual-priority MINs operating under weighted round-robin (WRR) and class-based weighted fair queuing (CBWFQ) scheduling disciplines have been reported insofar in the literature[1], we validated our simulator only against those that use strict priority scheduling (SP).

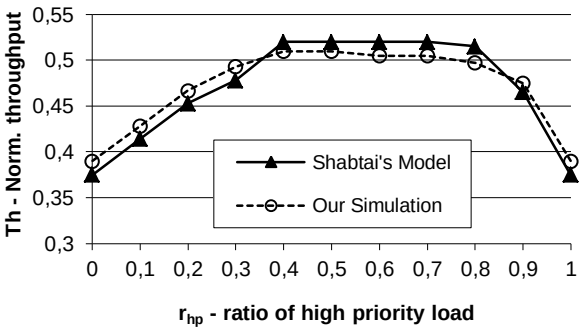

Fig. 3 Total normalized throughput of a dual-priority, single-buffered, 6-stage MIN.

More specifically, we compared our measurements against those obtained from Shabtai's Model reported in [24], and have found that both results were in close agreement (the maximum difference was only 3.8%). Fig. 3 illustrates this comparison, involving the *total normalized throughput* for all packets (both high- and low-priority) of a dual-priority, single-buffered, 6-stage MIN vs. the *ratio of high priority packets* under full *offered load* conditions.

B. Overall MIN throughput

Before examining the QoS offered to each priority class under the examined scheduling disciplines, we will present the simulation results regarding the effect that each scheduling algorithm has on the overall throughput of the MIN. Fig. 4a and Fig. 4b depict the *total normalized throughput* [th=$th(h)$+$th(l)$] of MIN using a dual-priority scheme when different scheduling algorithms are employed. In these diagrams, curves CBWFQ:R[X%]W[Y%] depict the *total normalized throughput* of MIN, when the CBWFQ scheduling algorithm was employed, the *ratio* of high priority offered load was set to X% and the weighting probability of servicing a *high priority queue* was configured to be Y%. Similarly, curves WRR:R[X%]TS[Z/W] show the performance evaluation of the same metric, employing the WRR scheduling discipline, when the *ratio* of high priority packets is again X% and the servicing cycles of each high priority queue is set to Z time-slots per totally W time-slots in every SE. Finally, curves PQ:R[X%] illustrate the *total normalized throughput* of MIN, employing the PQ scheduling algorithm, under a *ratio* of high priority offered load equal to X%.

In these diagrams we can observe that when the MIN operates under low and medium load ($\lambda \leq 0.6$), the overall MIN throughput is practically the same under all scheduling disciplines in both scenarios. Nevertheless, when the MIN

[1] notably, [7] reports on simulating the performance of a class-based weighted fair queuing system, but its focus lies on simulation quality and convergence, not network performance; additionally [21] reports on the performance of WRR over WiMax environments but the simulation design in this study is greatly affected by WiMax-related parameters such as topology, MAC layer intrinsics etc., and therefore its results are not directly comparable to the ones reported in this paper.

operates under high load ($\lambda \geq 0.7$), the WRR algorithm is found to provide the best overall *normalized throughput*, followed by CBWFQ, while PQ exhibits the minimum *normalized throughput* among the three algorithms. The improvement of WRR over PQ is up to 2.3% in the normal scenario (r_h=0.3), while in the extreme scenario (r_h=0.7) it reaches 6.3%; the corresponding improvements for CBWFQ are 1.9% and 3.9%. This performance improvement can be attributed to the fact that network resources are better exploited when using WRR and CBWFQ. This particularly applies to network buffers dedicated to low-priority queues within the SEs: under the strict priority mechanism, these buffers have decreased probability of transmitting the packets they hold, which in turn leads to increased probability of blockings, in the event that a new low-priority packet arrives at the corresponding SE. The *total normalized throughput* achieved by WRR is slightly superior to the one observed under CBWFQ, due to the synchronization on rotational servicing of high and low priority queues in all SEs, which reduces the probability of blocking due to having full buffers.

Since PQ is the "gold standard" for high priority packets, from the results depicted in Fig. 4a and Fig. 4b we can intuitively derive that there exists a tradeoff between choosing to offer high-priority packets better quality of service (i.e. employing PQ) and achieving optimal MIN performance (by utilizing either WRR or CBWFQ). In the following paragraphs, we will examine the QoS level delivered to packets of different priority classes under the three scheduling disciplines.

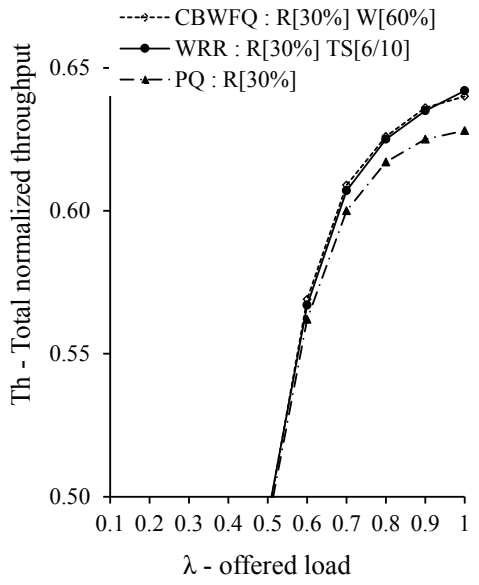

Fig. 4a Total normalized throughput through various queuing scheduling disciplines, normal scenario.

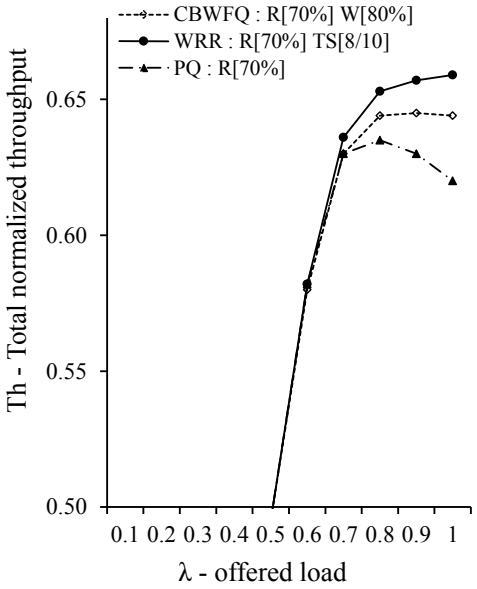

Fig. 4b Total normalized throughput through various queuing scheduling disciplines, extreme scenario.

C. Relative Normalized Throughput

Fig. 5a and Fig. 5b present the *relative normalized throughput* of a double-buffered, 6-stage MIN using a 2-class priority scheme under various scheduling disciplines. Since strict priority queueing (PQ) is the gold standard for high priority traffic, the *relative normalized throughput* of high priority packets was found to be at extremely high levels, approaching the optimal value [$RTh(h)_{max}$=1] of this performance factor, under full load traffic conditions, when the first scenario was employed [i.e. the offered load of high priority packets accounts to 30% of the overall traffic (Fig. 5a)]. Practically, under the above traffic classification the blocking events for high-priority packets were very rare.

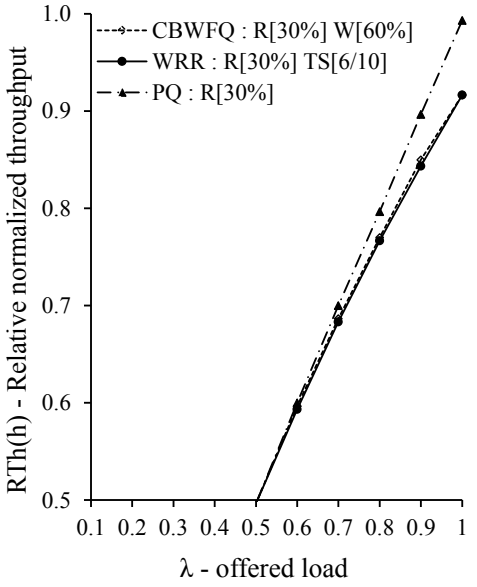

Fig. 5a Relative normalized throughput of high-priority packets through various queuing scheduling disciplines, normal scenario

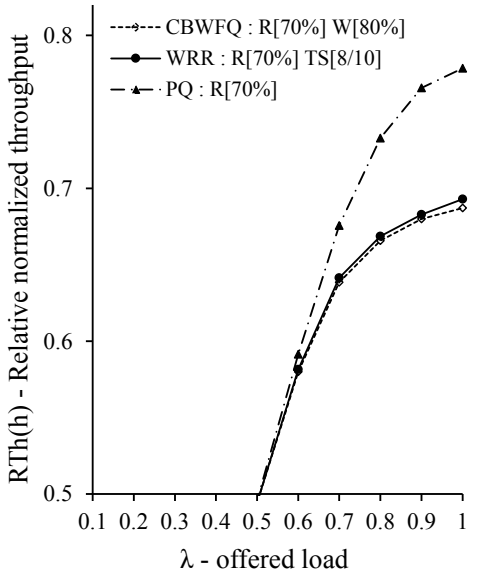

Fig. 5b Relative normalized throughput of high-priority packets through various queuing scheduling disciplines, extreme scenario

It is also worth noting that the *relative normalized throughput* of high priority packets remains at satisfactory levels $Rth(h)\approx 0.78$, even in the extreme scenario (Fig. 5b), in which the ratio of high priority packets was set to 70% of the total *input load*.

Comparing the CBWFQ and WRR scheduling disciplines, we observe that the curves referring to high priority traffic exhibit similar behavior. Recall that the rationale behind applying one of the two previously mentioned algorithms is to provide fair treatment for packets of all individual priority classes, avoiding thus a potential starvation of lower-priority queues. In these implementations, we noticed that the additional buffer space of high-priority packets was exploited only under very high input loads ($\lambda>0.7$). This phenomenon was more evident at the extreme scenario, where the proportion of high-priority offered load was set to the 70% of the overall traffic, under a process throttling 20% for servicing first the low-priority queues if they were not empty. However, when applying the CBWFQ or WRR algorithm, the throughput loss for high priority packets was found to be only up to 7.7% for the first scenario (Fig. 5a), whereas the corresponding loss at the second case study was again found to be at tolerable levels, approximately 11% (Fig. 5b) as compared to the case of using PQ scheduling (both maxima occur under full load traffic conditions i.e. $\lambda=1$).

Regarding the quality of service offered to low-priority packets (Fig. 6a and Fig. 6b), we can observe that the *relative normalized throughput* of these packets is considerably better in all configuration setups, when either the CBWFQ or WRR scheduling discipline was employed, as compared to the case of having applied the classical strict priority queuing (PQ) algorithm.

At the first scenario (Fig. 6a), where the proportion of low-priority offered load was set to 70% of the overall traffic and the weighting probability of servicing first the low-priority packets was configured to be 40%,the gain for low-priority traffic was approximately 11.2% under both CBWFQ and WRR scheduling disciplines.

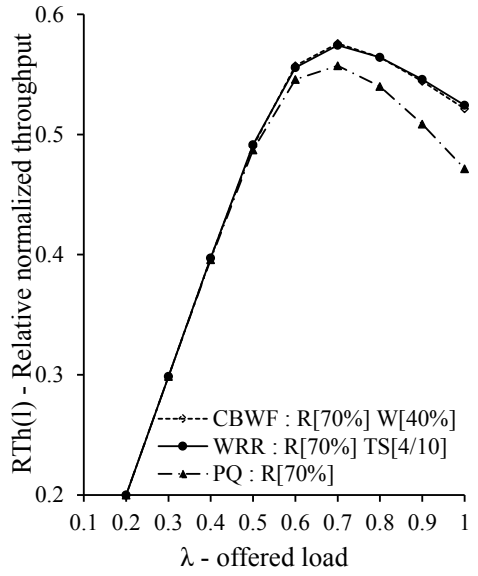

Fig. 6a Relative normalized throughput of low-priority packets through various queuing scheduling disciplines, normal scenario.

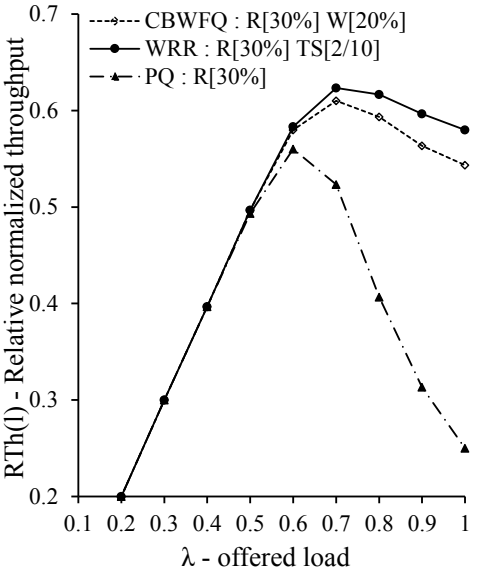

Fig. 6b Relative normalized throughput of low-priority packets through various queuing scheduling disciplines, extreme scenario.

On the other hand, regarding the second scenario (Fig. 6b), where the proportion of low-priority offered load was set to 30% of the overall traffic, we observed that the WRR scheduling algorithm had a small performance edge over CBWFQ. It is worth mentioning that the *relative normalized throughput* of low-priority traffic exhibits spectacular improvement rates under both scheduling mechanisms (132% and 117% respectively), as compared to the corresponding classical PQ scheduling. This performance gain is owing to the fact that low-priority queues have 20% probability to be

examined before the adjacent high-priority ones for packet forwarding, while in parallel the amount of high-priority traffic is great (70% of the overall traffic) and thus the probability of finding a high-priority queue empty is low. Finally, considering the above two fair queuing algorithms, it is also worth noting that the *normalized throughput* for low-priority packets is slightly superior under the WRR scheduling discipline as compared to CBWFQ, due to the synchronization on rotational servicing of high and low priority queues in all SEs, which reduces the probability of blocking due to having full buffers.

D. Normalized Delay

Fig. 7a and Fig. 7b illustrate the *normalized delay* for high-priority traffic under the above two testbed scenarios, where the *ratio* of high-priority packets is 30% and 70% respectively.

Regarding Fig. 7a, the *normalized delay* of high-priority traffic was found always to be less than 1.15, and thus close to the optimum value 1 under the PQ scheduling discipline, indicating rare blockings for high-priority packets and consequently lower exploitation of the second buffer-space allocated to queues of this particular priority class in all SEs. Considering the other two fair scheduling algorithms (CBWFQ and WRR), we noticed that *normalized delay* went up slightly reaching the values of 1.34 and 1.32 respectively, indicating better exploitation of resources (buffer spaces) under a tolerable level of the delay factor. We also observed that the WRR algorithm had a small performance edge as compared to the CBWFQ scheduling when the total *offered load* was greater than 50% (λ>0.5).

Regarding the second scenario (Fig. 7b), both CBWFQ and WRR algorithms were observed to exhibit identical behavior. Employing the above two fair algorithms, we found that the worst value for *normalized delay* metric of high-priority packets was approximately 1.78 and slightly better than the corresponding value obtained by the PQ algorithm implementation, which was just 1.83 under full load traffic conditions. However, when the input load did not exceed the 90% of its allowed maximum value (i.e. λ<0.9), high-priority queues were found to exhibit a smaller number of blocking events under the configuration of a strict priority queuing (PQ) mechanism providing thus better servicing-times (i.e. lower delay).

Similarly Fig. 8a and Fig. 8b present the *normalized delay* for low-priority traffic under the above two testbed scenarios, where the *ratio* of low-priority packets was 70% and 30% respectively. In both these configurations we found that both CBWFQ and WRR algorithms exhibit identical behavior in *normalized delay* of low-priority traffic. Contrary to the case *of normalized throughput* discussed in the previous subsection, we can observe in Fig. 8a that CBWFQ algorithm had a small performance edge over the corresponding WRR counterpart. In this case the worst values for the delay factor were found to be 1.93 and 1.97 under full load traffic conditions, when the WRR and CBWFQ scheduling algorithms were employed respectively. Since both scheduling disciplines are fair in serving all class-priority traffic the gains for low-priority traffic were found again to be considerable.

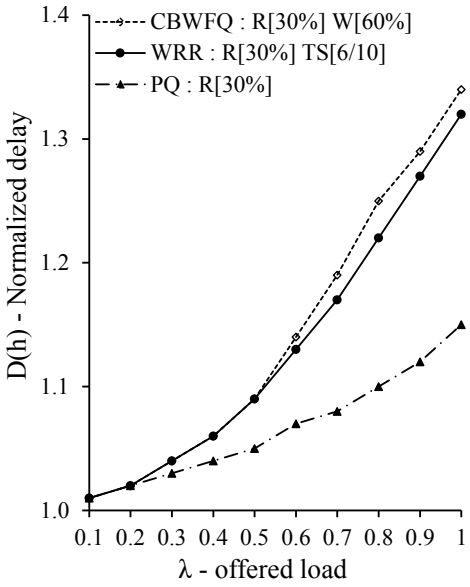

Fig. 7a Normalized delay of high-priority packets through various queuing scheduling disciplines, normal scenario.

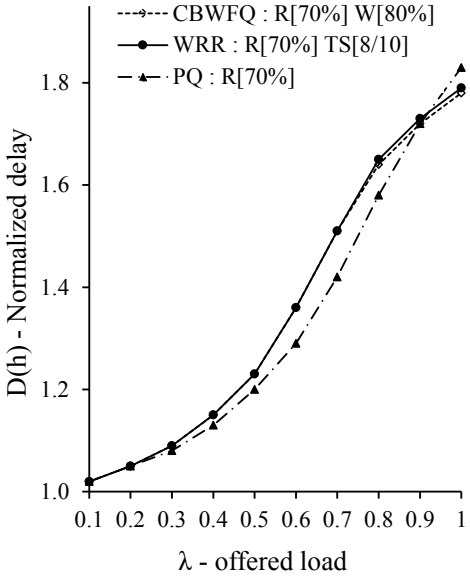

Fig. 7b Normalized delay of high-priority packets through various queuing scheduling disciplines, extreme scenario.

At the second scenario (Fig. 8b), where the amount of high-priority traffic is great (70% of the overall traffic) the *normalized delay* for low-priority traffic was found to reach higher values. More analytically, under the PQ scheduling discipline, the value of this metric was found to be 2.37, when the corresponding *normalized delay* for the other two fair algorithms was only 1.75 (gain 26.16%). Consequently, employing either the CBWFQ or WRR mechanism a considerable number of head-of-line blockings of low-priority packets that could be occurred under a strict priority queuing (PQ) scheduling configuration is prevented.

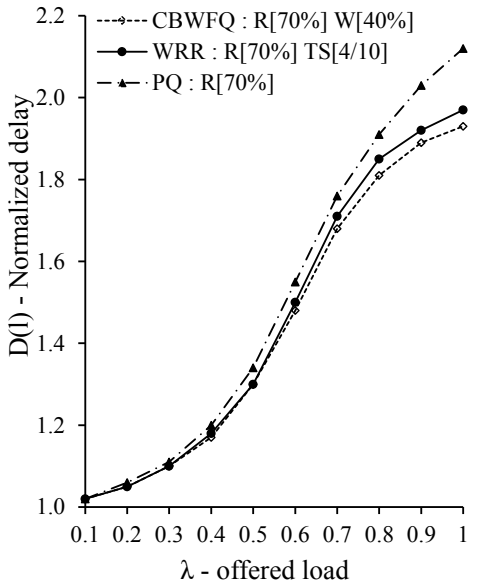

Fig. 8a Normalized delay of low-priority packets through various queuing scheduling disciplines, normal scenario.

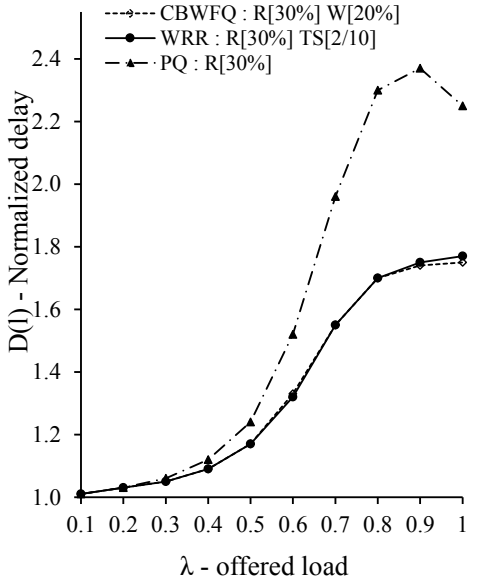

Fig. 8b Normalized delay of low-priority packets through various queuing scheduling disciplines, extreme scenario.

E. Universal Performance Factor

Fig. 9a and Fig. 9b show the *universal performance factor* for high-priority traffic under the two examined scenarios, where the *ratio* of high-priority packets is 30% and 70% respectively. In Fig. 9a we can notice that the combined performance metric (*Upf*) approaches its optimal value 0 under a strict priority queuing (PQ) implementation, since this discipline is the gold standard for high priority traffic. PQ begins to outperform CBWFQ or WRR for loads λ>0.6, since below this load the MIN has amble bandwidth available to optimally service high priority packets under all examined scheduling disciplines. We can also observe that in the first scenario the PQ algorithm had a greater performance edge over the other two fair scheduling disciplines as compared to the improvement it offers in the second scenario, since the *ratio* of high-priority *offered load* at this example was at low levels (only 30% of the overall traffic) and thus almost all high-priority packets were serviced within the minimum *packet delay*. On the other hand, when employing a fair queuing scheduling (CBWFQ or WRR) a small side-effect in the overall performance of high-priority traffic appears, which was however found to be tolerable at all examined configuration setups.

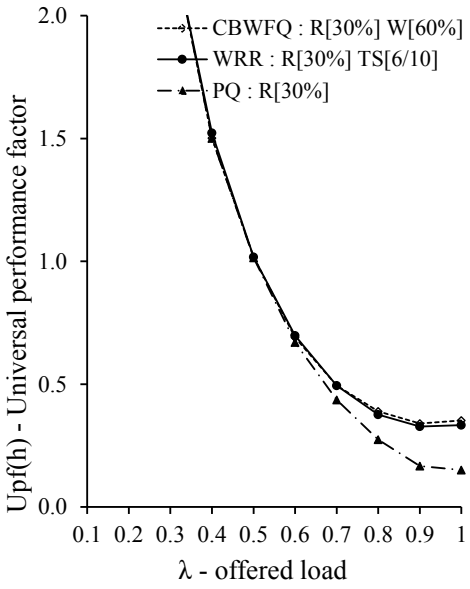

Fig. 9a Universal performance factor of high-priority packets through various queuing scheduling disciplines, normal scenario.

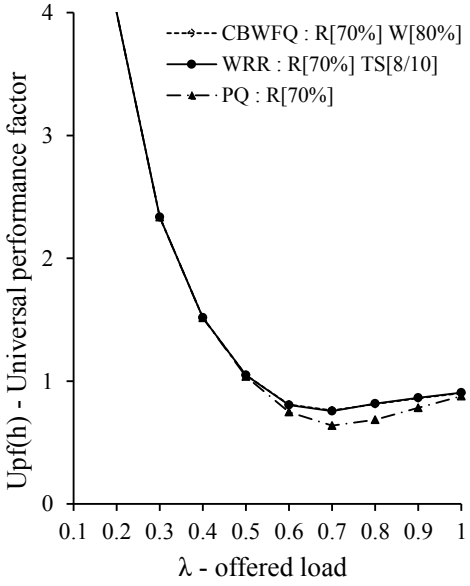

Fig. 9b Universal performance factor of high-priority packets through various queuing scheduling disciplines, extreme scenario.

Finally, regarding the low-priority traffic the gains in the overall performance were found to be spectacular when one of the two fair algorithms (CBWFQ or WRR) was employed. At the first scenario (Fig. 10a), where r_h was set to a value commonly observed in IP networks, we observed a small performance edge of CBWFQ algorithm over the WRR scheduling discipline, whereas at the second more extreme

case study, where the amount of high-priority traffic was great (70% of the overall traffic), the WRR algorithm appeared to have a small performance edge over CBWFQ (Fig. 10b). More analytically, employing one of the two fair queuing algorithms (CBWFQ or WRR), we observed that the overall performance gain at the first test bed example was approximately 16.2%, while at the second scenario the corresponding gain was found to be 64.5% -extremely high- under full *offered load* (λ=1) traffic conditions as compared to the classical PQ scheduling implementation.

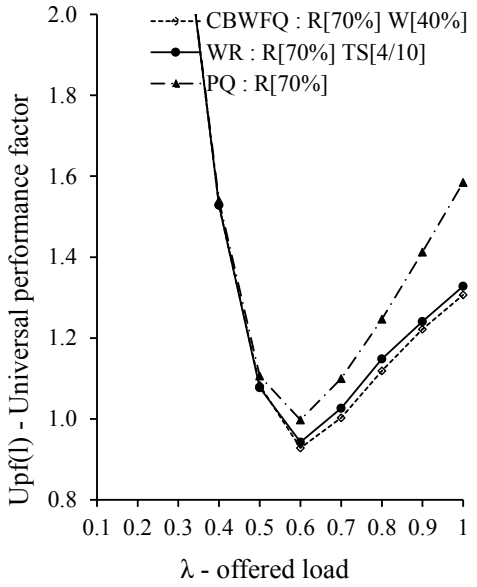

Fig. 10a Universal performance factor of low-priority packets through various queuing scheduling disciplines, normal scenario.

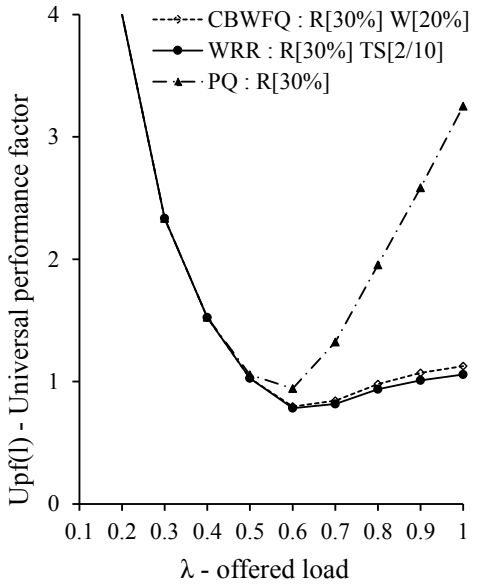

Fig. 10b Universal performance factor of low-priority packets through various queuing scheduling disciplines, extreme scenario.

F. Performance Metrics for a 10-Stage MIN

In order to validate that the results presented in the previous paragraphs hold for MINs of different sizes, we have conducted the same experiments for a 10-stage MIN. Fig. 11a presents the *relative normalized throughput* of low-priority packets as determined by this experiment, while Fig 11b illustrates the *normalized delay* of low priority packets. We can observe that the shape of the curves in Fig. 11a is identical to that of Fig. 6a (depicting the *relative normalized throughput* of low-priority packets for a 6-stage MIN), thus the behavior of the scheduling algorithms in the two MINs regarding this metric is very similar. All scheduling algorithms achieve smaller throughput for low-priority packets in the 10-stage MIN: this is due to the fact that each additional stage introduces an extra point that a blocking can occur, and the increased number of blockings lead to degraded throughput. The fact that low priority packet throughput drops when the MIN size increases has also been reported in other performance studies, e.g. [41].

Similarly, the shape of the curves in Fig. 11b is identical to that of Fig. 8a (illustrating the *normalized delay* of low priority packets for a 6-stage MIN), thus the behavior of the scheduling algorithms in the two MINs regarding this metric too is very similar. All scheduling algorithms exhibit higher *normalized delay* for low-priority packets in the 10-stage MIN, again owing to the increased probability of blockings in the extra stages. The finding that low-priority packet delay deteriorates when the number of MIN stages increases is consistent with the results of other works, e.g. [41].

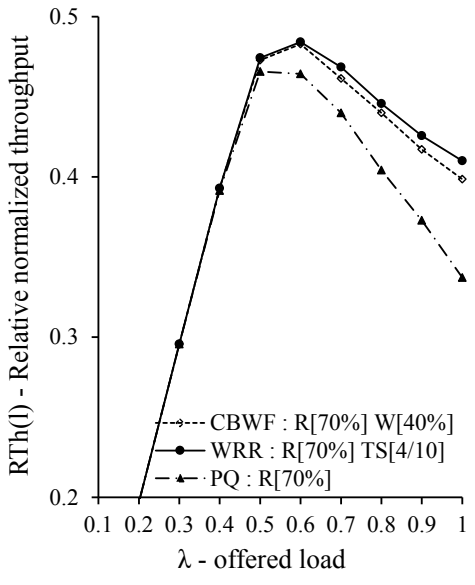

Fig. 11a Relative normalized throughput of low-priority packets through various queuing scheduling disciplines, normal scenario.

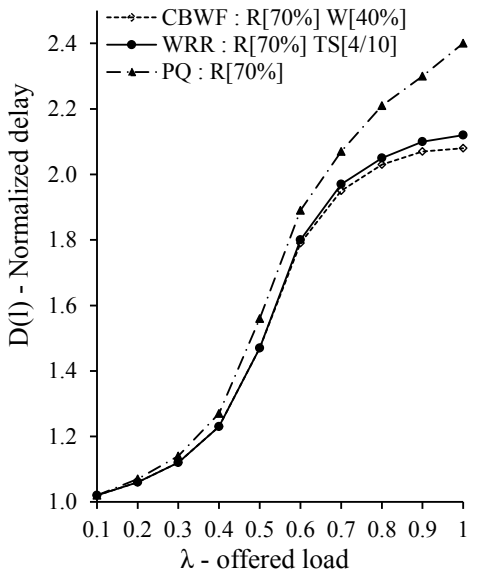

Fig. 11b Normalized delay of low-priority packets through various queuing scheduling disciplines, normal scenario.

We have thoroughly analyzed the behavior of the 10-stage MIN, considering all performance metrics examined in subsections C-F of section V, we do not include here the diagrams for conciseness purposes: all results indicate that the behavior of the scheduling algorithms is very similar among the 6-stage and the 10-stage MINs in all cases. The performance metrics for high-priority packets (throughput, delay, universal performance factor) are practically unaffected when the MIN size increases, but the performance metrics for low priority packets deteriorate, for the reasons explained above. These findings indicate that we can generalize our conclusions to apply to MINs of any size.

VI. Conclusions

In this paper, we have presented a comparative performance study of three scheduling algorithms, namely strict priority queuing (PQ), class-based weighed fair queuing (CBWFQ), and weighed round-robin (WRR) through simulation experiments. More analytically, the performance evaluation conducted in a double-buffered, 6-stage Multistage Interconnection Network that natively supported 2-class priority traffic, considering additionally a 10-stage MIN to validate that the conclusions drawn are valid for MINs of different sizes.

The performance metrics obtained through simulations re-ascertained the belief that the strict priority algorithm is the "gold standard" for high priority traffic, but on the other hand it was found to considerably degrade the quality of service offered to low-priority packets, and in parallel deteriorated the overall throughput of MIN. Regarding the other two examined fair algorithms, it is noticed that both CBWFQ and WRR mechanisms exhibited similar behavior in the quality of service offered to each particular priority class traffic, with both of them considerably improving the QoS offered to low-priority traffic. For instance, the *relative normalized throughput* of low-priority traffic was found to be spectacularly improved (117% under CBWFQ and 132% under WRR) as compared to the case of having applied the PQ discipline, under the scenario of setting λ_h=70% and w_h=80%. Table I summarizes the results presented in section V for the normal scenario, presenting the improvements and deteriorations of performance metrics when CBWFQ and WRR is employed, as compared to the corresponding metrics of PQ (in Table I, we consider 100% to be the value of the metric under PQ).In this table, only loads $\lambda \geq 0.3$ are taken into account, since for smaller loads the MIN has amble resources to serve all packets close to optimally. Similarly, Table II summarizes the results for the extreme scenario.

TABLE I
Performance of CBWFQ and WRR (Baseline: PQ), Normal Scenario

Priority class	Metric	CBWFQ			WRR		
		Mean	Min	Max	Mean	Min	Max
High	Throughput	97.7%	92.3%	100.0%	97.3%	92.3%	100.0%
Low	Throughput	103.6%	100.0%	110.6%	103.7%	100.0%	11.2%
High	Delay	108.6%	101.0%	116.5%	107.5%	100.1%	114.8%
Low	Delay	95.43%	91.04%	99.1%	96.59%	92.9%	99.1%

TABLE II
Performance of CBWFQ and WRR (Baseline: PQ), Extreme Scenario

Priority class	Metric	CBWFQ			WRR		
		Mean	Min	Max	Mean	Min	Max
High	Throughput	95.0%	88.3%	100.0%	95.3%	89.0%	100.0%
Low	Throughput	133.0%	100.0%	217.3%	137.3%	100.0%	232.0%
High	Delay	102.3%	97.3%	106.34%	102.5%	97.8%	106.3%
Low	Delay	85.3%	73.4%	99.1%	85.4%	73.8%	99.1%

Since the WRR discipline was based conceptually on the round robin scheduling, it could be simply implemented at hardware level. Such an approach is considered to be fast: it simplifies the translation of the QoS requirements into a number of time-slots serviced by a particular class queue. On the other hand, the CBWFQ discipline provides support for user-defined traffic classes, but the probabilistic mechanism which was applied individually in each SE was more complicated. Consequently, a Class-Based Queuing (CBQ) version of WRR scheduling discipline called CB-WRR [1], where a network operator will be allowed to define traffic classes and apply parameters, such as bandwidth and queue-limits, could be adopted as a solution to QoS guarantees in the MIN technology.

Future work will focus on examining other load configurations, including hot-spot and burst loads.

References

[1] H. Aouad. Transport de Flux Temps Réels dans un Réseau IP Mobile. PhD Thesis, l'Ecole Nationale Supérieure des Télécommunications. http://www.ethesis.net/flux/Hazar_AOUAD.pdf, accessed January 19, 2014.

[2] Avaya Inc. Avaya "Automatic QoS Technical Configuration Guide for the ERS 4500, 5000, Avaya BCM 50, 450, Avaya CS 1000, Avaya CS 2100 and Avaya SRG 50", http://support.avaya.com/css/P8/documents/100123842, accessed January19, 2014.

[3] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017,

http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html
[4] Dax networks. Dax Managed Gigabit Switch Dx-5024GME. http://www.daxnetworks.com/products/switches-1/layer-2-switches/dx-5024gme.html#, accessed January 19, 2014.
[5] Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", Journal of Internetworking Research and Experience, V1, N1, pp 3-26, September 1990.
[6] Elwalid , D. Mitra, "Design of generalized processor sharing schedulers which statistically multiplex heterogeneous QoS classes", Proceedings of 18th Annual Joint Conference of IEEE Computer and Communications Societies (INFOCOM '99), pp. 1220 – 1230, 1999.
[7] M., Fischer, D., B. Masi, J., Shortle. Simulating the performance of a class-based weighted fair queuing system. Proceedings of the 2008 Winter Simulation Conference, pp. 2901-2908, IEEE press.
[8] J. Garofalakis and E. Stergiou. "An analytical performance model for multistage interconnection networks with blocking", in proceedings of the CNSR 2008, May 2008. Proc. pp 373–381.
[9] J., Gozdecki, A., Jajszczyk, and R., Stankiewicz. Quality of Service Terminology in IP Networks. IEEE Communications Magazine, pp. 153-159, March 2003.
[10] Guo. SRR, "An O(1) time complexity packet scheduler for flows in multi-service packet networks", IEEE/ACM trans. Networking, 12(6):pp. 1144–1155, Dec. 2004.
[11] K. Gupta, L. O. Barbosa, N.D. Georganas, "Switching modules for ATM switching systems and their interconnection networks", Computer Networks and ISDN Systems, Volume 26, Issue 4, pp. 433-445, December 1993.
[12] Hewlet-Packard. HP ProCurve Secure Router 7000dl Series. Available at http://h17007.www1.hp.com/us/en/products/routers/HP_A7000dl_Router_Series/index.aspx?jumpid=reg_r1002_usen_c-001_title_r0001 accessed January19, 2014.
[13] S. Hiyama, Y. Nishino and I. Sasase, "Multistage Interconnection Multicast ATM Switch with Exclusive Routes for Delay-Sensitive and Loss-Sensitive Cells", Journal of High Speed Networks - JHSN , vol. 15, no. 2, pp. 131-155, 2006.
[14] G.U. Hwang and F. Ishizaki, "Packet level performance analysis of a packet scheduler exploiting multiuser diversity", Telecommunication Systems vol. 45, pp. 249–258, 2010
[15] M. Ilyas and M. A. Syed, "An efficient multistage switching node architecture for broadband ISDNs", Telecommunication Systems vol. 10 pp. 229–241, 1998.
[16] J. Liebeherr, E. Yilmaz., "Workconserving vs. Non-workconserving Packet Scheduling", An Issue Revisited. Seventh International Workshop on Quality of Service (IWQoS '99), 1999, pp. 248 – 256, 1999.
[17] Mahgoub and C.J. Huang, "A novel scheme to improve fault-tolerant capabilities of multistage interconnection networks", Telecommunication Systems vol. 10, pp. 45–66, 1998.
[18] C. McKillen, S. Sezer, X. Yng, High performanceservice time-stamp-computing for WFQ IP packet scheduling, Proceeding of Emerging VLSI Technologies and Architectures, Karlsruhe, Germany, 65-70 (2009)
[19] Nortel Networks. Nortel Ethernet Switch 460/470 Overview — System Configuration. http://support.avaya.com/css/P8/documents/100099692, accessed January 19, 2014.
[20] Prabhakar, N. McKeown. On the speedup required for combined input- and output-queued switching. Automatica, Volume 35, Issue 12,pp.1909-1920, December 1999.
[21] A., H., Rashwan, H., M. ElBadawy, H., H., Ali. Comparative Assessments for Different WiMAX Scheduling Algorithms. Proceedings of the World Congress on Engineering and Computer Science Vol I, 2009.
[22] RFCs for Differentiated Services in the Internet Architecture http://www.ietf.org/rfc/rfc2474.txt http://www.ietf.org/rfc/rfc2475.txt
[23] RFCs for Integrated Services in the Internet Architecture http://www.ietf.org/rfc/rfc1613.txt| http://www.ietf.org/rfc/rfc2212.txt http://www.ietf.org/rfc/rfc2215.txt
[24] G. Shabtai, I. Cidon, and M. Sidi, "Two priority buffered multistage interconnection networks", Journal of High Speed Networks, pp.131–155, 2006
[25] G. Shabtai, I. Cidon, and M. Sidi, "Two Priority Buffered Multistage Interconnection Networks", IEEE High Performance Switching and Routing Conference HPSR'04 pp.75-79, 2004
[26] J. Shortle, M. Fisher. Approximation for a two-class weighted fair queuing discipline. Performance Evaluation 67 (2010) 946-958.
[27] M. Shreedhar, G. Varghese, "Efficient fair queuing using deficit round-robin", IEEE/ACM Transactions on Networking, vol. 4 issue 3, pp. 375 – 385, June 1996.
[28] T. Soumiya, K. Nakamichi, S. Kakuma, T. Hatano, and A. Hakata, The large capacity ATM backbone switch "FETEX-150 ESP", Comput. Netw. 31(6) , pp. 603–615, 1999.
[29] E. Stergiou, G. Garofalakis, "Performance evaluation for multistage interconnection networks servicing unicast and multicast traffic (by partial operation)" Proceedings of the Performance Evaluation of Computer and Telecommunication Systems (SPECTS'09), IEEE Press, pp. 311 - 318 , July, 2009.
[30] T.H. Theimer, E. P. Rathgeb and M.N. Huber. "Performance Analysis of Buffered Banyan Networks", IEEE Transactions on Communications, vol. 39, no. 2, pp. 269-277, February 1991.
[31] J. Torrellas and Z. Zhang, "The performance of the cedar multistage switching network", IEEE Trans. Parallel Distrib. Syst. 8(4) , pp. 321–336, 1997.
[32] E.S.H. Tse, "Switch fabric architecture analysis for a scalable bi-directionally reconfigurable IP router", Journal of Systems Architecture, EUROMICRO J. 50(1) , pp. 35–60, 2004.
[33] Tutsch, G.Hommel. "Comparing Switch and Buffer Sizes of Multistage Interconnection Networks in Case of Multicast Traffic", Procs. of the High Performance Computing Symposium, (HPC 2002); San Diego, SCS, pp. 300-305, 2002.
[34] Tutsch and G. Hommel. "Multilayer Multistage Interconnection Networks", Proceedings of 2003 Design, Analysis, and Simulation of Distributed Systems (DASD'03). Orlando, USA, pp. 155-162, 2003.
[35] D.C. Vasiliadis, G.E. Rizos, and C. Vassilakis. "Performance Analysis of blocking Banyan Swithces", Procs. of CISSE 06, December, 2006.
[36] D.C. Vasiliadis, G.E. Rizos, C. Vassilakis, and E.Glavas. "Performance evaluation of two-priority network schema for single-buffered Delta Network", Procs. of IEEE PIMRC' 07, 2007.
[37] D.C. Vasiliadis, G.E. Rizos, C. Vassilakis. "Improving Performance of Finite-buffered Blocking Delta Networks with 2-class Priority Routing through Asymmetric-sized Buffer Queues", Proceedings of the Fourth Advanced International Conference on Telecommunications AICT08, IEEE Press, 2008.
[38] D.C. Vasiliadis, G.E. Rizos, C. Vassilakis. "Class-Based Weighted Fair Queuing Scheduling on Dual-Priority Delta Networks", Journal of Computer Networks and Communications, Article ID 859694, 13 pagesdoi:10.1155/2012/859694, Volume 2012 (2012)
[39] D.C. Vasiliadis, G.E. Rizos, and C. Vassilakis. "Routing and Performance Evaluation of Dual Priority Delta Networks under Hotspot Environment", Proceedings of the First International Conference on Advances in Future Internet AFIN09, IEEE Press, pp. 24-30, 2009.
[40] D.C. Vasiliadis, G.E. Rizos, C. Vassilakis. Performance Study of Multilayered Multistage Interconnection Networks under Hotspot Traffic Conditions. Journal of Computer Systems, Networks, and Communications, doi:10.1155/2010/403056, Volume 2010 (2010).
[41] D.C. Vasiliadis, G.E. Rizos, C. Vassilakis. "Modelling and performance study of finite-buffered blocking multistage interconnection networks supporting natively 2-class priority routing traffic", Journal ofNetworkandComputerApplications36, pp. 723–737, 2013.
[42] A. Waldspurger, W. E. Weihl, "Lottery Scheduling: Flexible Proportional-Share Resource Management", In Proceedings of symposim on Operating System Design and Implementation, November 1994.
[43] Wikipedia. Round-robin scheduling. http://en.wikipedia.org/wiki/Round-robin_scheduling
[44] Xin. Li, L. Mhamdi, J. Liu, K. Pun, M. Hamdi, "Architectures of Internet Switches and Routers", In High-performance Packet Switching Architectures, I. Elhanany and M. Hamdi (eds), ISBN: 1-84628-273-X, Springer-Verlag London Limited 2007.
[45] L. Yang, C. Pan, E. Zhang and H.Y. Liu, "A New Class of Priority-based Weighted Fair Scheduling Algorithm", 2012 International Conference on Medical Physics and Biomedical Engineering, Vol. 33, pp. 942–948, 2012.
[46] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, D. RSVP: a new resource ReSerVation Protocol. IEEE Network, vol. 7(5), pp. 8-18, September 1993.

Dr. Dimitris Vasiliadis is currently head of the Network Operations Center (NOC) of Technological Educational Institute of Epirus, a node of the Greek Research and Technology Network (GRNET). He is also a research fellow at University of Peloponnese, Greece and a member of IEEE Computer Society. He has participated as partner in several Greek and European IT and Telematics Projects. He has published more than 40 articles and has been a PC member and referee in various International Journals and Conferences. His research interests include Performance Analysis of Networking and Computer Systems, Computer Networks and Protocols, Security of Networks, Telematics, QoS and New Services.

Dr. George Rizos has received his PhD degree from the University of Peloponnese in Greece. He is currently a System Administrator at the Network Operations Center (NOC) of Technological Educational Institute (T.E.I.) of Epirus, a node of the Greek Research and Technology Network (GRNET). He has participated as partner in several Greek and European IT and Telematics Projects. He has published several papers. His research interests are focused on Wireless Communications, Ad hoc Networks and Protocols, Multimedia, and Quality of Networking Services.

Dr. Costas Vassilakis is currently an Associate Professor in the Department of Informatics and Telecommunications, University of Peloponnese, Greece. He has published over 130 papers in international scientific journals and conferences and has participated in more than 20 European and national research and development projects. He has been a PC member and referee in several International Journals and Conferences. His research interests include Semantic Web Technologies and Applications, Service-oriented Architectures, Distributed Systems, Infrastructure Performance, E-government and E-commerce, VR-Systems and Visualization.