Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 859694, 13 pages
doi:10.1155/2012/859694

Research Article

Class-Based Weighted Fair Queuing Scheduling on Dual-Priority Delta Networks

D. C. Vasiliadis,[1,2] G. E. Rizos,[1,2] and C. Vassilakis[1]

[1] *Department of Computer Science and Technology, University of Peloponnese, 22100 Tripolis, Greece*
[2] *Technological Educational Institute of Epirus, 47100 Arta, Greece*

Correspondence should be addressed to D. C. Vasiliadis, dvas@uop.gr

Received 28 October 2011; Accepted 23 January 2012

Academic Editor: Maode Ma

Contemporary networks accommodate handling of multiple priorities, aiming to provide suitable QoS levels to different traffic classes. In the presence of multiple priorities, a scheduling algorithm is employed to select each time the next packet to transmit over the data link. Class-based Weighted Fair Queuing (CBWFQ) scheduling and its variations is widely used as a scheduling technique, since it is easy to implement and prevents the low-priority queues from being completely neglected during periods of high-priority traffic. By using this scheduling, low-priority queues have the opportunity to transmit packets even though the high-priority queues are not empty. In this work, the modeling, analysis and performance evaluation of a single-buffered, dual-priority multistage interconnection network (MIN) operating under the CBWFQ scheduling policy is presented. Performance evaluation is conducted through simulation, and the performance measures obtained can be valuable assets for MIN designers, in order to minimize the overall deployment costs and delivering efficient systems.

1. Introduction

During the last decade, we have witnessed a dramatic increase in both network speeds and the amount of network traffic. In order to provide high quality-of-service (QoS) in today's high-speed networks, different priorities are assigned to packets entering the networks, and packet scheduling algorithms are employed to select each time the next packet to transmit over the data link. To this end, a number of packet scheduling algorithms have been proposed, with the most prominent ones including strict priority queuing [1], round-robin [2] and its variations (e.g., weighted round-robin [3, 4], deficit round-robin [5], smoothed round-robin [6]), generalized processor sharing (GPS) [7], weighted fair queuing (P-GPS) [8], class-based weighted fair queuing [9], virtual clock [10], and self-clocked fair queuing [11]. In a number of works (e.g., [12–14]), packets enter the MIN without a priority (as opposed to the previous approaches where the where priorities are assigned to packets before they enter the MIN), and the MIN internally prioritizes packets aiming either to offload the most heavily loaded queues and reduce blockings [12] or avoid crosstalk in optical MINs ([13, 14]); in essence, however, only the priority source changes (internal versus externally defined), while for selecting the most prominent packet for forwarding, one of the previously listed algorithms is applied.

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. 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 [6] (since they affect the memory and the processing required to implement the algorithm, resp.) 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.

Among the algorithms described above, strict-priority queuing (i.e., servicing lower priority packets only when higher-priority ones are not waiting to be serviced), weighted round robin (i.e., assigning a portion of the available bandwidth to each priority queue), and class-based weighted fair

queuing (i.e., having N data flows currently active, with weights $w_1, w_2, \ldots, w_N$, data flow i will achieve an average data rate of $R*w_i/(w_1+w_2+\cdots+w_N)$, where R is the data link rate) [9] have been adopted by the industry and implemented in most commercial products (e.g., [15–21]) 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 and processing power requirements, and (d) network administrators find them easy to understand and configure.

Regarding the network switch internal architecture, multistage interconnection networks (MINs) with crossbar switching elements (SEs) are frequently proposed for interconnecting processors and memory modules in parallel multiprocessor systems [22–24] and have also recently been identified as an efficient interconnection network for communication structures such as gigabit Ethernet switches, terabit routers, and ATM switches [25–27]. Significant advantages of MINs include their low cost/performance ratio and their ability to route multiple communication tasks concurrently. MINs with the Banyan [28] property are proposed to connect a large number of processors to establish a multiprocessor system; they have also received considerable interest in the development of packet-switched networks. Non-Banyan MINs are, in general, more expensive than Banyan networks and more complex than control.

In the current literature, the performance of multipriority MINs under the strict priority queuing algorithm has been studied extensively through both analytical methods and simulation experiments (e.g., [29–34]), considering various buffer sizes (mainly buffers of sizes 1, 2, and 4), buffer size allocation to different priority classes (symmetric versus asymmetric [30]), arrival processes (e.g., uniform versus bursty [35]), traffic patterns (e.g., uniform versus hotspot [4, 36, 37]; unicast versus multicast [38, 39]), and internal MIN architectures (e.g., single-layer versus multilayer [40]). These studies have shown that under high 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.

Using class-based weighted fair queuing as a packet scheduling algorithm instead of strict-priority queuing appears as a plausible solution for providing better QoS to low-priority packets under increased network load since one of the goals of this scheduling technique is to increase fairness, giving low-priority queues the opportunity to transmit packets even though the high-priority queues are not empty. Class-based weighted fair queuing overcomes some limitations of weighted round-robin, namely, the fact that it cannot guarantee fair link sharing and the need to know the mean packet size of each connection in advance [41]. Insofar, however, there are no studies to quantify (a) the gains obtained for low-priority packets (and conversely the losses incurred for high-priority packets) by employing the class-based weighted fair queuing packet scheduling algorithm and (b) the effect of the individual queue weight assignment to the overall performance of the multistage interconnection network and the QoS offered to packets of different priority classes.

In this paper, a simulation-based performance evaluation for a single-buffered MIN natively supporting two priority classes and employing the class-based weighted fair queuing packet scheduling algorithm is presented. Moreover, analytical equations have been derived from the new queuing modeling based on the one-clock history consideration. In this performance evaluation, we calculate the QoS offered to packets of different priority classes, under high network loads and under different ratios of high-/low-priority packets within the overall network traffic. We also study the effect of queue weight assignment in the QoS offered to packets of different priorities.

The rest of this paper is organized as follows: in Section 2 we present the dual priority MIN and give details on its operation and the class-based weighted fair queuing packet scheduling algorithm. In Section 3, we present the analytical equations for the MIN, extending Mun's [42] 3-state model to a 6-state one for improving its accuracy. In Sections 4 and 5, we present the performance metrics and the simulation results, respectively, while in Section 6 conclusions are drawn and future work is outlined.

2. Dual-Priority MIN and the Class-Based Weighted Fair Queuing Scheduling Algorithm

A Multistage Interconnection Network (MIN) can be defined as a network 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. Its main characteristics are its topology, routing algorithm, switching strategy, and flow control mechanism.

All types of blocking Multistage Interconnection Networks (Delta Networks [43], Omega Networks [44], and Generalized Cube Networks [45]) with the Banyan property which is defined in [28] are characterized by the fact that there is exactly a unique path from each source (input) to each sink (output). Banyan MINs are multistage self-routing switching fabrics. Consequently, each SE of kth stage, where $k = 1, \ldots, n$, can decide in which output port to route a packet, depending on the corresponding kth bit of the destination address.

A typical configuration of an $(N \times N)$ Delta Network is depicted in Figure 1. In order to support priority handling, each SE has two transmission queues per link, accommodated in two (logical) buffers, with one queue dedicated to high-priority packets and the other dedicated to low-priority ones. In this paper, we consider a dual-priority Multistage Interconnection Network with the Banyan property that operates under the following assumptions.

(i) The network clock cycle consists of two phases. In the first phase, flow control information passes through the network from the last stage to the first one. In the second phase, packets flow from one stage to the next in accordance to the flow control information.

(ii) The arrival process of each input of the network is a simple Bernoulli process, that is, the probability that

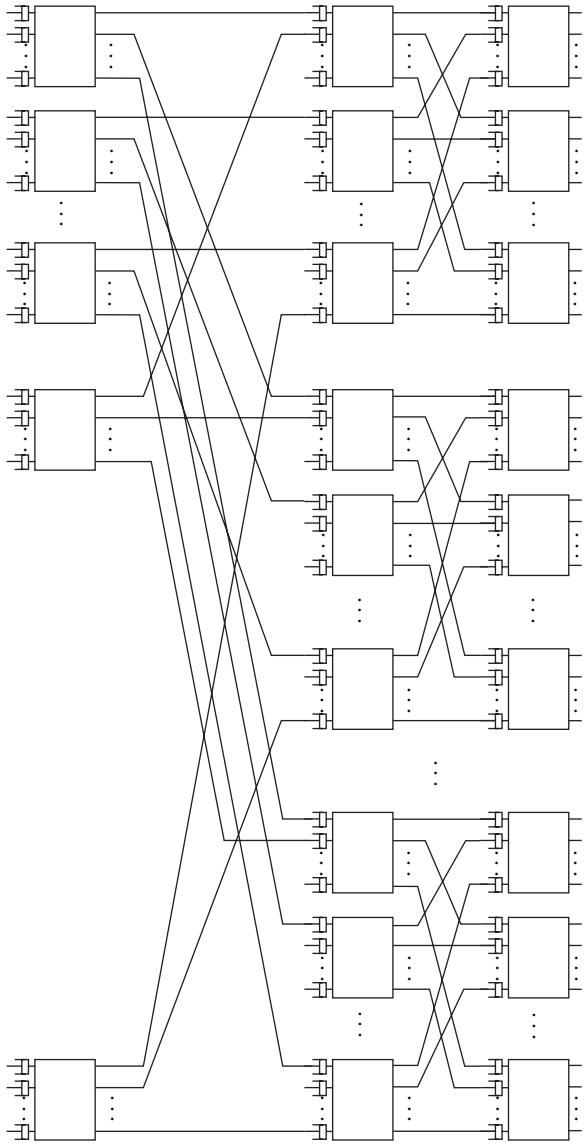

Figure 1: A 3-stage Delta Network.

a packet arrives within a clock cycle is constant and the arrivals are independent of each other. 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 $\lambda = \lambda_h + \lambda_l$.

(iii) Under the dual-priority mechanism, when applications or architectural modules enter a packet to the network, they specify its priority, designating it either as high or low. The criteria for priority selection may stem from the nature of packet data (e.g., packets containing streaming media data can be designated as high-priority while FTP data can be characterized as low-priority), from protocol intrinsics (e.g., TCP out-of-band/expedited data versus normal connection data [46]), or from properties of the interconnected system architecture elements.

(iv) A high-/low-priority packet arriving at the first stage ($k = 1$) is discarded if the high-/low-riority buffer of the corresponding SE is full, respectively.

(v) A high-/low-priority packet is blocked at a stage if the destination high-/low-priority buffer at the next stage is full, respectively.

(vi) Both high- and low-priority packets are uniformly distributed across all destinations, and each high-/low-priority queue uses an FIFO policy for all output ports.

(vii) 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.

(viii) Upon reception, packets are first classified according to their priority and are then assigned to the queue specifically dedicated to the particular priority (Figure 2).

(ix) 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 nonempty 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_k is the weight assigned to queue k [9]. This is analogous to lottery scheduling used in operating systems [47]. We note here that the class-based weighted fair queuing algorithm considered in this paper is *work conserving*, that is, a packet is always transmitted when there is traffic waiting, as opposed to *nonwork conserving* algorithms which do not transmit a packet if the queue whose turn is to transmit a packet is found to be empty [48]. 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.

(x) 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), there is a conflict. Conflict resolution in a single-priority mechanism operates under the following scheme: one packet will be accepted at random and the other will be blocked by means of upstream control signals. In a dual-priority mechanism, the class-based weighted fair queuing algorithm determines which class of two buffer queues is serviced by the SE processor.

The priority class of each packet is indicated through a priority bit in the packet header, thus it suffices for the SE to read the header in order to make a decision on which packet to store and which one to block.

(xi) All SEs have deterministic service time.

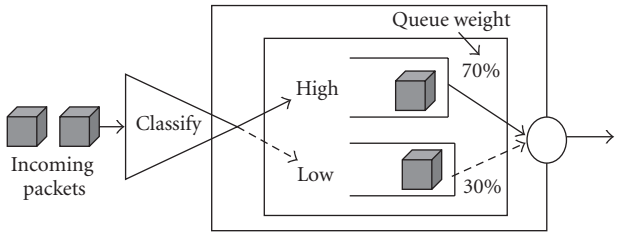

Figure 2: Class-based weighted fair queuing algorithm.

(xii) Finally, all packets in input ports contain both the data to be transferred and the routing tag. In order to achieve synchronously operating SEs, the MIN is internally clocked. As soon as packets reach a destination port they are removed from the MIN, so packets cannot be blocked at the last stage.

3. Analytical Equations for the Dual-Priority MIN

Our analysis introduces a novel model, which considers not only the current state of the associated buffer but also the previous one. Based on the one clock history consideration, we enhance Mun's [42] three states model with a six-state buffer model, which is described in the following paragraphs.

3.1. State Notations for c-Class Priority Queues. Since the proposed model is exemplified in a single-buffered configuration the buffer state will be either empty "0" or full "1" at each clock cycle. Taking into account the history of covering one clock cycle, the following states are examined.

(i) *State* "00^c": c-class priority buffer was empty at the beginning of the previous clock cycle and it is also empty at beginning of the current clock cycle.

(ii) *State* "01^c": c-class priority buffer was empty at the beginning of the previous clock cycle, while it contains a new c-class priority packet at the current clock cycle (a new packet arrived).

(iii) *State* "10^c": c-class priority buffer had a packet at the previous clock cycle, while it contains no packet at the current clock cycle (the packet was transmitted and no new packet was received).

(iv) *State* "$11n^c$": c-class buffer had a packet at the previous clock cycle and has a new one at the current clock cycle (the previous one was successfully transmitted and the new packet was just received).

(v) *State* "$11b^c$": c-class buffer had a packet at the previous clock cycle and has the same packet at the current clock cycle; an attempt was made to transmit the packet during the previous clock cycle but it failed due to blocking.

(vi) *State* "$11w^c$": c-class buffer had a packet at the previous clock cycle and has the same packet waiting at the current clock cycle, because the conjugate priority queue ($c\sim$-class priority queue) had also a packet ready to be transmitted in the previous clock cycle, and the bandwidth allocation algorithm selected the packet in the $c\sim$-class priority queue for transmission. Within a switching element SE, the conjugate of the high-priority queue is the low-priority queue of the same element SE, and vice versa.

3.2. Definitions for c-Class Priority Queues. The following variables are defined in order to develop an analytical model. In all definitions, SE(k) denotes an SE at *stage* k of the MIN

(i) $P_{00}(k,t)^c$ is the probability that a c-class priority buffer of SE(k) is empty at both $(t-1)$th and tth network cycles.

(ii) $P_{01}(k,t)^c$ is the probability that a c-class priority buffer of SE(k) is empty at $(t-1)$th network cycle and has a new c-class priority packet at tth network cycle.

(iii) $P_{10}(k,t)^c$ is the probability that a c-class priority buffer of SE(k) has a c-class priority packet at $(t-1)$th network cycle and is empty at tth network cycle.

(iv) $P_{11n}(k,t)^c$ is the probability that a c-class priority buffer of SE(k) has a packet at $(t-1)$th network cycle and has a new one at tth network cycle.

(v) $P_{11b}(k,t)^c$ is the probability that a c-class priority buffer of SE(k) has a packet at $(t-1)$th network cycle and still has the same packet at tth network cycle, as the packet could not be transmitted due to blocking.

(vi) $P_{11w}(k,t)^c$ is the probability that a c-class priority buffer of SE(k) has a packet at $(t-1)$th network cycle and still has the same packet at tth network cycle, as the packet could not be transmitted because the conjugate priority queue ($c\sim$-class priority queue) had also a packet ready to be transmitted at $(t-1)$th network cycle, and the bandwidth allocation algorithm selected the packet in the $c\sim$-class priority queue for transmission.

(vii) $q(k,t)^c$ is the probability that a c-class priority packet is ready to be sent into a buffer of SE(k) at tth network cycle (i.e., a c-class priority packet will be transmitted by an SE($k-1$) to SE(k)).

(viii) $r_{01}(k,t)^c$ is the probability that a c-class priority packet in a buffer of SE(k) is ready to move forward during the tth network cycle, given that the buffer is in "01^c" state.

(ix) $r_{11n}(k,t)^c$ is the probability that a c-class priority packet in a buffer of SE(k) is ready to move forward during the tth network cycle, given that the buffer is in "$11n^c$" state.

(x) $r_{11b}(k,t)^c$ is the probability that a c-class priority packet in a buffer of SE(k) is ready to move forward during the tth network cycle, given that the buffer is in "$11b^c$" state.

(xi) $r_{11w}(k,t)^c$ is the probability that a c-class priority packet in a buffer of SE(k) is ready to move forward

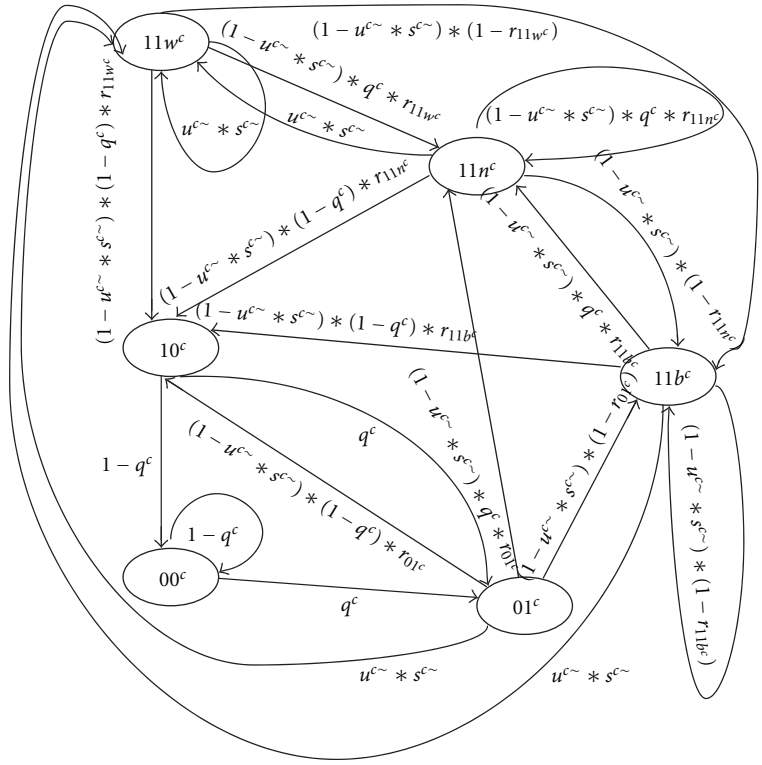

Figure 3: A state transition diagram of a c-class priority buffer of SE(k).

during the tth network cycle, given that the buffer is in "$11w^c$" state.

3.3. Mathematical Analysis for c-Class Priority Queues. The following equations, derived from the state transition diagram in Figure 3, represent the state transition probabilities of c-class priority queues as clock cycles advance.

The probability that a c-class priority buffer of SE(k) was empty at the $(t-1)$th network cycle is $P_{00}(k,t-1)^c + P_{10}(k,t-1)^c$. Therefore, the probability that a c-class priority buffer of SE(k) is empty both at the current tth and previous $(t-1)$th network cycles is the probability that the SE(k) was empty at the previous $(t-1)$th network cycle multiplied by the probability $[1-q(k,t-1)^c]$ of no c-class priority packet was ready to be forwarded to SE(k) during the previous network cycle (the two facts are statistically independent, thus the probability that both are true is equal to the product of the individual probabilities). Formally, this probability $P_{00}(k,t)^c$ can be expressed by

$$P_{00}(k,t)^c = \left[1-q(k,t-1)^c\right] * \left[P_{00}(k,t-1)^c + P_{10}(k,t-1)^c\right]. \quad (1)$$

The probability that a c-class priority buffer of SE(k) was empty at the $(t-1)$th network cycle and a new c-class priority packet has arrived at the current tth network cycle is the probability that the SE(k) was empty at the $(t-1)$th network cycle (which is equal to $P_{00}(k,t-1)^c + P_{10}(k,t-1)^c$) multiplied by the probability $q(k,t-1)^c$ that a new c-class priority packet was ready to be transmitted to SE(k) during the $(t-1)$th network cycle. Formally, this probability $P_{01}(k,t)^c$ can be expressed by

$$P_{01}(k,t)^c = q(k,t-1)^c * \left[P_{00}(k,t-1)^c + P_{10}(k,t-1)^c\right]. \quad (2)$$

The case that a c-class priority buffer of SE(k) was full at the $(t-1)$th network cycle but is empty during the tth network cycle effectively requires the following two facts to be true: (a) a c-class priority buffer of SE(k) was full at the $(t-1)$th network cycle and the c-class priority packet was successfully transmitted and (b) no c-class priority packet was received during the $(t-1)$th network cycle to replace the transmitted c-class priority packet into the buffer. The probability for fact (a) is equal to the product of the following two probabilities: (i) the probability that the SE processor was not occupied by the packet of the adjacent queue of SE(k), which is just $[1-U(k,t-1)^{c\sim} * s^{c\sim}]$, where $U(k,t-1)^{c\sim}$ expresses the probability that a packet exists in the adjacent $c\sim$-class priority queue of SE(k) during network cycle $t-1$ and $s^{c\sim}$ denotes the service rate given by the class-based weighted fair queuing this $c\sim$-class priority queue; (ii) $[r_{01}(k,t-1)^c * P_{01}(k,t-1)^c + r_{11n}(k,t-1)^c * P_{11n}(k,t-1)^c + r_{11b}(k,t-1)^c * P_{11b}(k,t-1)^c + r_{11w}(k,t-1)^c * P_{11w}(k,t-1)^c]$; this probability is computed by considering all cases that during the network cycle $t-1$ the SE had a c-class priority packet in its buffer and multiplying the probability of each state by the corresponding probability that the packet was successfully transmitted to a next-stage SE. Finally, the probability of fact (b), that is, that no c-class

priority packet was ready to be transmitted to SE(k) during the previous network cycle is equal to $[1 - q(k,t-1)^c]$. Con-sequently, the probability $P_{10}(k,t)^c$ can be computed by the following formula:

$$\begin{aligned} P_{10}(k,t)^c = {} & \left[1-U(k,t-1)^{c\sim} * s^{c\sim}\right] * \left[1-q(k,t-1)^c\right] \\ & * \left[r_{01}(k,t-1)^c * P_{01}(k,t-1)^c + r_{11n}(k,t-1)^c \right. \\ & * P_{11n}(k,t-1)^c + r_{11b}(k,t-1)^c \\ & * P_{11b}(k,t-1)^c + r_{11w}(k,t-1)^c \\ & \left. * P_{11w}(k,t-1)^c\right]. \end{aligned} \tag{3}$$

The probability that a c-class priority buffer of SE(k) had a packet at the $(t-1)$th network cycle and has also a new one (different than the previous; the case of having the same packet in the buffer is addressed in the next paragraphs) at the tth network cycle is the probability that the SE processor was not occupied by the packet of the adjacent queue of SE(k) at the $(t-1)$th network cycle, which is just $[1-U(k,t-1)^{c\sim} * s^{c\sim}]$, multiplied firstly by the probability of having a ready c-class priority packet to move forward at the previous $(t-1)$th network cycle [which is equal to $r_{01}(k,t-1)^c * P_{01}(k,t-1)^c + r_{11n}(k,t-1)^c * P_{11n}(k,t-1)^c + r_{11b}(k,t-1)^c * P_{11b}(k,t-1)^c + r_{11w}(k,t-1)^c * P_{11w}(k,t-1)^c$] and multiplied secondly by $q(k,t-1)^c$, that is, the probability that a c-class priority packet was ready to be transmitted to SE(k) during the previous network cycle. Formally, this probability $P_{11n}(k,t)^c$ can be expressed by

$$\begin{aligned} P_{11n}(k,t)^c = {} & \left[1-U(k,t-1)^{c\sim} * s^{c\sim}\right] * q(k,t-1)^c \\ & * \left[r_{01}(k,t-1)^c * P_{01}(k,t-1)^c + r_{11n}(k,t-1)^c \right. \\ & * P_{11n}(k,t-1)^c + r_{11b}(k,t-1)^c \\ & * P_{11b}(k,t-1)^c + r_{11w}(k,t-1)^c \\ & \left. * P_{11w}(k,t-1)^c\right]. \end{aligned} \tag{4}$$

The next case that should be considered is when a c-class priority buffer of SE(k) had a packet at the $(t-1)$th network cycle and still contains the same packet blocked at the tth network cycle. This occurs when the packet in the c-class priority buffer of SE(k) was ready to move forward at the $(t-1)$th network cycle, but it was blocked (not forwarded) during that cycle, due to a blocking event—either (a) the associated c-class priority buffer of the next stage SE was already full due to another blocking, or (b) buffer space was available at stage $k+1$ but it was occupied by a second packet of the current stage contending for the same c-class priority buffer during the process of forwarding. The probability for this case can be formally defined as

$$\begin{aligned} P_{11b}(k,t)^c = {} & \left[1 - U(k,t-1)^{c\sim} * s^{c\sim}\right] \\ & * \left\{\left[1 - r_{01}(k,t-1)^c\right] * P_{01}(k,t-1)^c \right. \\ & + \left[1 - r_{11n}(k,t-1)^c\right] * P_{11n}(k,t-1)^c \\ & + \left[1 - r_{11b}(k,t-1)^c\right] * P_{11b}(k,t-1)^c \\ & \left. + \left[1 - r_{11w}(k,t-1)^c\right] * P_{11w}(k,t-1)^c\right\}. \end{aligned} \tag{5}$$

The final case that should be considered is when a c-class priority buffer of SE(k) had a packet at the $(t-1)$th network cycle and still contains the same packet waiting to get access to SE processor at the tth network cycle. This occurs when the packet in the c-class priority buffer of SE(k) remained in a wait-state during that cycle, due to the fact that the SE processor was occupied by the packet of the adjacent queue of SE(k); this probability is $[U(k,t-1)^{c\sim} * s^{c\sim}]$. Consequently, the probability for this case can be formally defined as

$$\begin{aligned} P_{11w}(k,t)^c = {} & U(k,t-1)^{c\sim} * s^{c\sim} \\ & * \left[P_{01}(k,t-1)^c + P_{11n}(k,t-1)^c \right. \\ & \left. + P_{11b}(k,t-1)^c + P_{11w}(k,t-1)^c\right]. \end{aligned} \tag{6}$$

The factor $U(k,t-1)^{c\sim}$ can be evaluated by the following equation:

$$\begin{aligned} U(k,t-1)^{c\sim} = {} & r_{01}(k,t-1)^{c\sim} * P_{01}(k,t-1)^{c\sim} \\ & + r_{11n}(k,t-1)^{c\sim} * P_{11n}(k,t-1)^{c\sim} \\ & + r_{11b}(k,t-1)^{c\sim} * P_{11b}(k,t-1)^{c\sim} \\ & + r_{11w}(k,t-1)^{c\sim} * P_{11w}(k,t-1)^{c\sim}. \end{aligned} \tag{7}$$

The factor $[1 - U(k,t-1)^{c\sim} * s^{c\sim}]$ appearing in the previous equations effectively manifests that the corresponding states may only be reached if the adjacent $c\sim$-class priority queues do not use the SE processor: this holds because the pertinent states may be reached if only a packet is transmitted from a c-class priority queue, where an empty or waiting $c\sim$-class priority queue is a prerequisite for such a transmission to occur.

Adding (1)–(6), both left, and right-hand sides are equal to 1, validating thus that all possible cases are covered; indeed $P_{00}(k,t)^c + P_{01}(k,t)^c + P_{10}(k,t)^c + P_{11n}(k,t)^c + P_{11b}(k,t)^c + P_{11w}(k,t)^c = 1$ and $P_{00}(k,t-1)^c + P_{01}(k,t-1)^c + P_{10}(k,t-1)^c + P_{11n}(k,t-1)^c + P_{11b}(k,t-1)^c + P_{11w}(k,t-1)^c = 1$.

The system of equations presented in the previous paragraphs extends the ones presented in other works (e.g., [49]) by considering the state and transitions occurring within an additional clock cycle. All previous works were based on a three-state model. This enhancement with a six-state buffer model can improve the accuracy of the performance parameters calculation (*throughput* and *delay*). The simulation

presented in following sections takes into account all the above-presented dependencies among the queues of each SE(k) of the MIN. In our future work, we intend to have additionally a closed form solution providing thus an analytical model for single-buffered MINs incorporating the class-based weighted fair queuing algorithm on a dual-priority scheme.

4. Performance Evaluation Metrics for Dual-Priority MINs

The two most important network performance factors, namely, *packet throughput* and *delay* are evaluated and analyzed in this section. The *Universal performance factor* introduced in [50], which combines the above two metrics into a single one, is also applied. 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. This is not necessarily true for all application classes, for example, for batch data transfers *throughput* is more important, whereas for streaming media the *delay* must be optimized. In order to evaluate the performance of an $(N \times N)$ MIN, the following metrics are used. Let Th and D be the *normalized throughput* and *normalized delay* of an 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:

$$\mathrm{RTh}(h) = \frac{\mathrm{Th}(h)}{r_h}. \tag{8}$$

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

$$\mathrm{RTh}(l) = \frac{\mathrm{Th}(l)}{r_l}. \tag{9}$$

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 [50]: the performance of an 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

$$\mathrm{Upf} = \sqrt{w_d * D^2 + w_{\mathrm{th}} * \frac{1}{\mathrm{Th}^2}}, \tag{10}$$

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 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, (10) can be replaced by

$$\mathrm{Upf} = \sqrt{w_d * \left(\frac{D - D^{\min}}{D^{\min}}\right)^2 + w_{\mathrm{th}} * \left(\frac{\mathrm{Th}^{\max} - \mathrm{Th}}{\mathrm{Th}}\right)^2}, \tag{11}$$

where $D^{\min}$ is the minimum value of *normalized packet delay* (D) and $\mathrm{Th}^{\max}$ is the maximum value of *normalized throughput*. Consistently to (10), when the *universal performance factor* Upf, as computed by (11) is close to 0, the performance an 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 (11) are normalized, $D^{\min} = \mathrm{Th}^{\max} = 1$, thus the equation can be simplified to

$$\mathrm{Upf} = \sqrt{w_d * (D-1)^2 + w_{\mathrm{th}} * \left(\frac{1 - \mathrm{Th}}{\mathrm{Th}}\right)^2}. \tag{12}$$

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:

$$\mathrm{Upf}(p) = \sqrt{w_d * (D(p)-1)^2 + w_{\mathrm{th}} * \left(\frac{1 - \mathrm{RTh}(p)}{\mathrm{RTh}(p)}\right)^2}, \tag{13}$$

where $p = \{h, l\}$ stands for high- and low-priority traffic, respectively.

In the remaining of this paper, we will consider both weight factors of equal importance, setting thus $w_d = w_{\mathrm{th}} = 1$.

Finally, we list the major parameters affecting the performance of examining dual-priority MIN.

Buffer size (b) is the maximum number of packets that an input buffer of an SE can hold. In our paper, we consider a single-buffered ($b = 1$) MINs.

Offered load (λ) is the steady-state fixed probability of arriving packets at each queue on inputs. In our simulation, the λ is assumed to be $\lambda = 0.65$ or 1.

Ratio of high-priority offered load (r_h), where $r_h = \lambda_h/\lambda$. In our study, r_h is assumed to be $r_h = 0.20$ or 0.30.

Service rate of high-priority packets (s^h) is the percentage rate of processor dedicated to high-priority packets by the class-based weighted fair queuing. In our simulation, s^h is assumed to be $s^h = 0, 0.1, 0.2, \ldots, 0.9, 1$.

Network size n, where $n = \log_2 N$, is the number of stages of an $(N \times N)$ MIN. In our simulation, n is assumed to be $n = 6$.

5. Simulation and Performance Results

In this paper, we developed a special simulator in C++, capable of handling dual-priority MINs using the class-based weighted fair queuing. Each (2×2) SE was modeled by four nonshared buffer queues, where buffer operation was based on the first come first serviced principle; the first two buffer queues for high-priority packets (one per incoming link), and the other two for low-priority ones.

Performance evaluation was conducted by using simulation experiments. Within the simulator, several parameters such as the *buffer-length,* the *number of input and output ports*, the *ratio of high-priority offered load,* the *service rate of high-priority packets,* and *the traffic shape* was considered.

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.

5.1. Simulator Validation. To validate our simulator, we compared the results obtained from our simulator against the results reported in other works, selecting among them the ones considered most accurate. Figure 4 shows the *normalized throughput* of a single-buffered, single-priority MIN with 6 stages as a function of the probability of arrivals for the three classical models [42, 49, 51] and our simulation.

All models are very accurate at low loads. The accuracy reduces as input load increases. In particular, when input load approaches the network maximum throughput, the accuracy of Jenq's model is insufficient. One of the reasons is the fact that many packets are blocked mainly at the network first stages at high traffic rates. Thus, Mun introduced a "blocked" state to his model to improve accuracy. Theimer's model considers the dependencies between the two buffers of an SE; this has lead to further improvement in accuracy and, therefore, Theimer's model is considered the most accurate insofar. Our simulation was also tested by comparing the results of Theimer's model with those of our simulation experiments, which were found to be in close agreement (differences are less than 1%).

5.2. Overall MIN Performance. Before examining the QoS offered to each priority class under different settings of the queue weights in CBWFQ, we will present the simulation results regarding the effect of queue weight setting to the overall performance of the MIN.

Figure 5 depicts the *total normalized throughput* [th = th(h) + th(l)] of an MIN using a dual-priority scheme versus the bandwidth dedicated to high-priority packets by the class-based weighted fair queuing. In the diagram, curve high-$X(\lambda = y)$ depicts the *total normalized throughput* of a 2-class priority, single-buffered 6-stage MIN, when the service ratio of high-priority packets is X% and offered load is y. We can notice here that the gains on *total normalized throughput* of a dual-priority scheme for a 6-stage, single-buffered MIN using the class-based weighted fair queuing algorithm versus the strict priority queuing mechanism are considerable. The performance of the strict priority queuing mechanism is effectively represented by the last value of each curve: if the weight of the high-priority queue is set to 1, then low-priority packets are served only when no high-priority packets are available, which is exactly the behavior of the strict priority queuing mechanism.

It is obvious that when offering greater servicing rates to low-priority queues, the *total normalized throughput* increases (except for the case of High-30 ($\lambda = 1$) where the

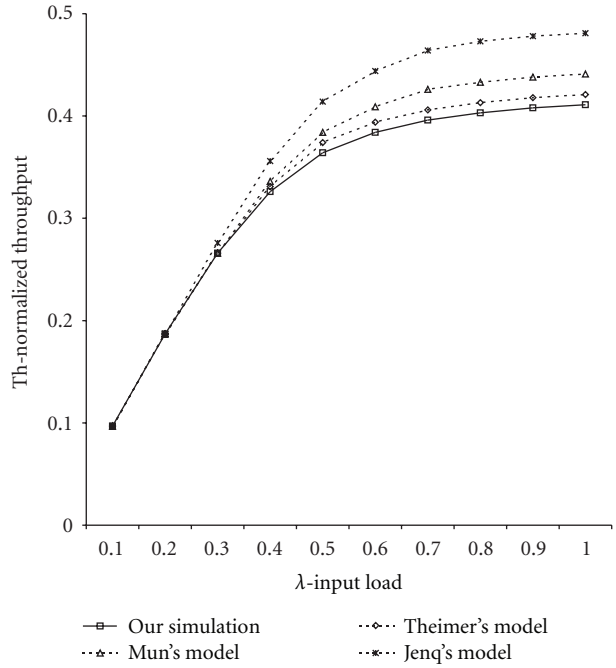

Figure 4: Normalized throughput of a single buffered 6-stage MIN.

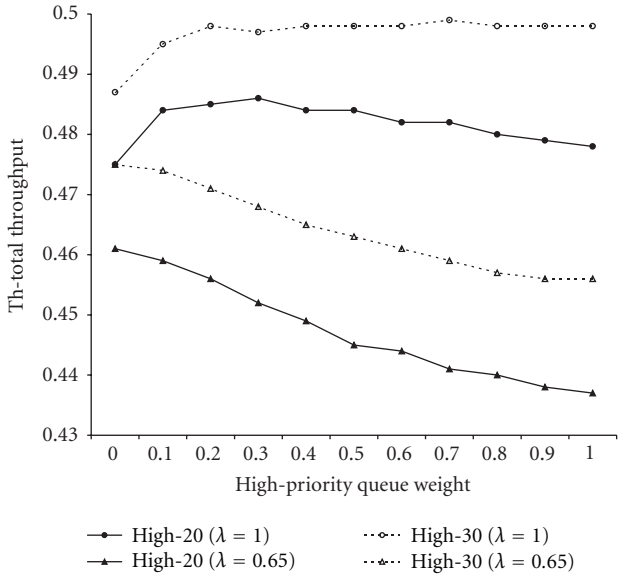

Figure 5: MIN throughput under varying high-priority queue weights.

performance remains at the same level) because the network resources are better exploited. 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. Nevertheless, the primary goal of classifying the packets into two priority classes is to provide better QoS to high-priority ones. This goal can simply be achieved when the weight of the high-priority queue for CBWFQ algorithm is set to a value greater than the anticipated load of high-priority packets. The exact setting of this parameter can be determined by balancing between the factors of achieving optimal overall network performance and delivering better QoS to high-priority packets. The QoS level delivered to packets of different priority classes under the CBWFQ algorithm is discussed in the following paragraphs.

5.3. Dual-Priority MINs Performance under Full-Load Traffic Conditions. In this subsection, we examine the QoS offered to packets of different priorities when the MIN is fully loaded ($\lambda = 1$). Figure 6 illustrates the *relative normalized throughput* for high- and low-priority packets under varying high-priority queue weights, and considering high-priority packet ratios of 20% and 30%. In this diagram, we can observe that—expectedly—when the high-priority queue weight increases, high-priority packets are offered better quality of service, while the QoS offered to low-priority packets drops. The leftmost part of the x-axis, where the high-priority queue weight is less than the ratio of high-priority packets in the network, is not bound to be used, since within that part-high-priority packets are offered worse quality of service than low-priority ones. Further increasing the high-priority queue weight up to 0.7 delivers an improvement of 30–42% for high-priority packets, whereas the corresponding deterioration for low-priority packets is much lower, ranging from 12% to 20%. For the last portion of the curves (high-priority queue weight between 0.7 and 1), the benefits for the high-priority packets is small (between 7.5% and 11.6%) and similar are the losses for low-priority packets (between 5.8 and 12%).

Note that since the diagram depicts the *relative normalized throughput metric* (which is normalized by the ratio of packets of the corresponding priority in the total load), a higher value in the diagram does not necessarily indicate *higher number of packets*, but merely the fact that the network handles packets more efficiently. Consequently, the fact that curve Low-80 crosses over curve Low-70 at high-priority queue weight ≈ 65% is interpreted that before this point low-priority packets in a 30/70 ratio are handled more efficiently than low-priority packets in a 20/80 ratio, whereas beyond this point the situation is reversed.

Figure 7 illustrates the *normalized delay* for high- and low-priority packets under varying high-priority queue weights, and considering high-priority packet ratios of 20% and 30%. Again, as the high-priority queue weight increases, high-priority packets are served faster, to the expense of the low-priority packets' delay. The overall variations in the delay, at least in the range 0.3–1.0 for the high-priority queue weight, are small (less than 12%), mainly due to the fact that the MIN considered in this paper is single-buffered, and single-buffered MINs tend to exhibit low values in delay, to the having however lower throughput and higher number of dropped packets [29, 30, 52]. The crossover of lines Low-80 and Low-70 at high-priority queue weight ≈ 70% is explained similarly to the case of the *relative normalized throughput*, discussed above.

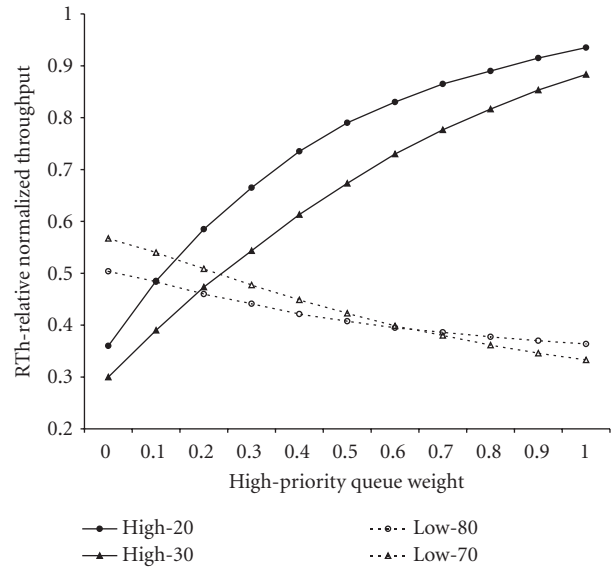

Figure 6: Normalized throughput for different priority classes under varying high-priority queue weights and full load.

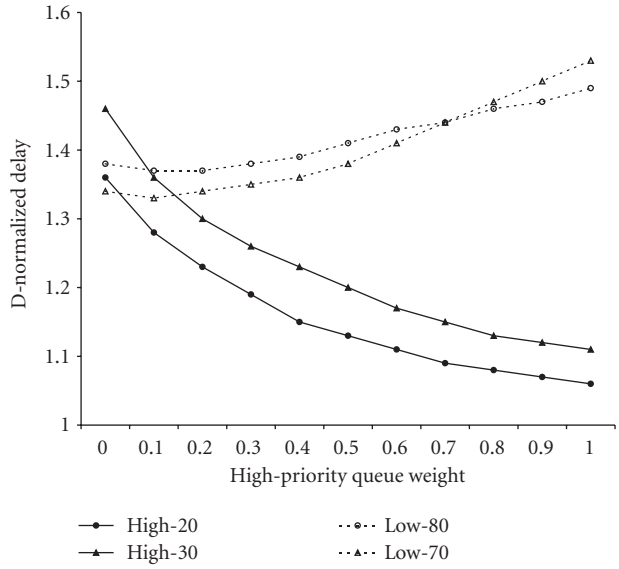

Figure 7: Normalized delay for different priority classes under varying high-priority queue weights and full load.

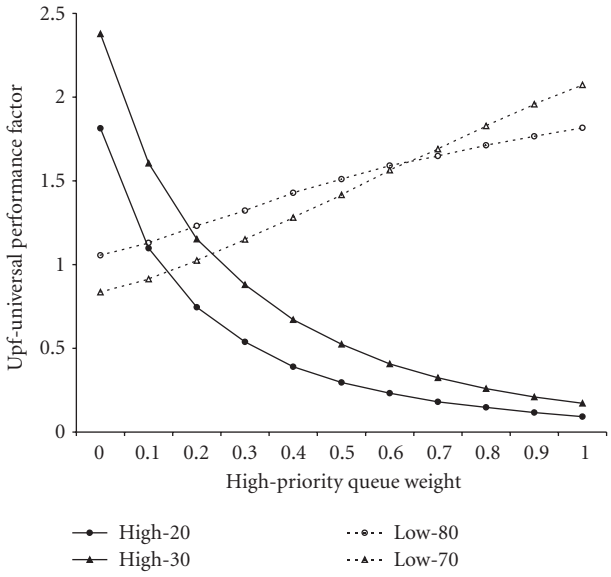

Figure 8: Universal performance factor for different priority classes under varying high-priority queue weights and full load.

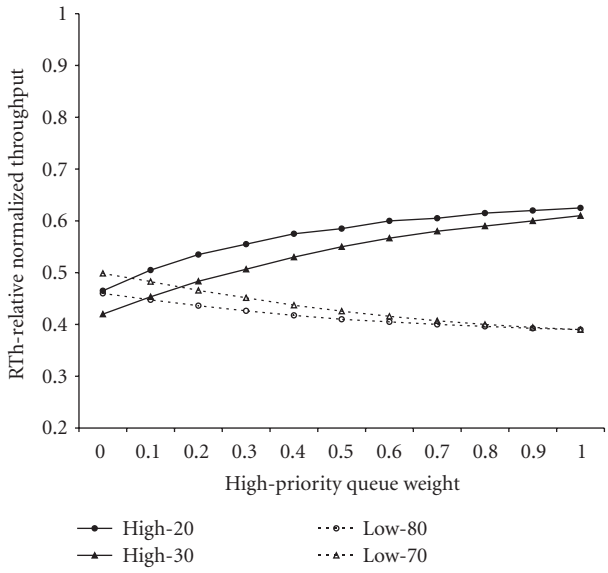

Figure 9: Normalized throughput for different priority classes under varying high-priority queue weights and high load.

Finally, Figure 8 depicts the *universal performance factor* (Upf) for different priority classes under varying high-priority queue weights and two high/low packet ratios (20/80 and 30/70). Since the individual performance factors (throughput and delay) combined in Upf evolve along a specific pattern (i.e., high-priority packets are served better as the high-priority queue weight increases while the inverse holds for low-priority packets), the same pattern is exhibited by the Upf too: its value drops (i.e., improves) for high-priority packets as the high-priority queue weight increases, while for low-priority packets its value rises (i.e., deteriorates) as the high-priority queue weight increases.

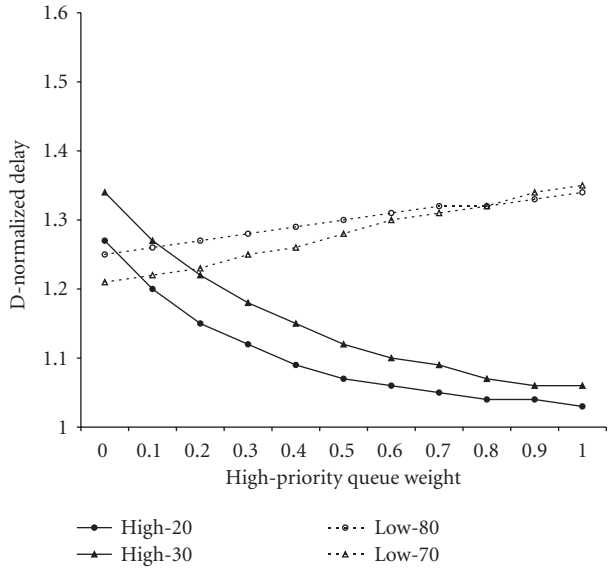

Figure 10: Normalized delay for different priority classes under varying high-priority queue weights and high load.

5.4. Dual-Priority MINs Performance under High Network Load. In this subsection, we examine the QoS offered to packets of different priorities when the MIN operates under high load, that is, the packet arrival probability λ is equal to 65% (approximately 2/3 of the full load). Figure 9 illustrates the *relative normalized throughput* for high- and low-priority packets under varying high-priority queue weights, and considering high-priority packet ratios of 20% and 30%. The trends of the curves are similar to the case of the full load (Figure 6), but the absolute values are smaller, since the *offered load* is smaller too. The improvement observed for high-priority packets when increasing the high-priority queue weight from 0.3 to 0.7 ranges from 9.0% to 14.5%, while in the full-load case, the corresponding improvement ranged from 30% to 42%. The smaller improvement is owing to the decreased network load, due to which high-priority packets are offered an increased quality of service, even for low values of high-priority queue weight, and, therefore, the margins for improvement are more limited. Similarly, the deterioration in the low-priority packets' throughput is limited, ranging from 6.2% to 9.8% (12% to 20% in the full load case). For the last portion of the curves (high-priority queue weight between 0.7 and 1), both the gains of high-priority packets and the losses for low-priority ones are less than 5% in all cases.

Figure 10 presents the *normalized delay* for different priority classes under varying high-priority queue weights and high load. When increasing the high-priority queue weight from 0.3 to 0.7, the delay for high-priority packets is improved between 6% and 8%, while the respective deterioration for low-priority packets ranges between 3% and 5%. The variations are small because, similarly to the

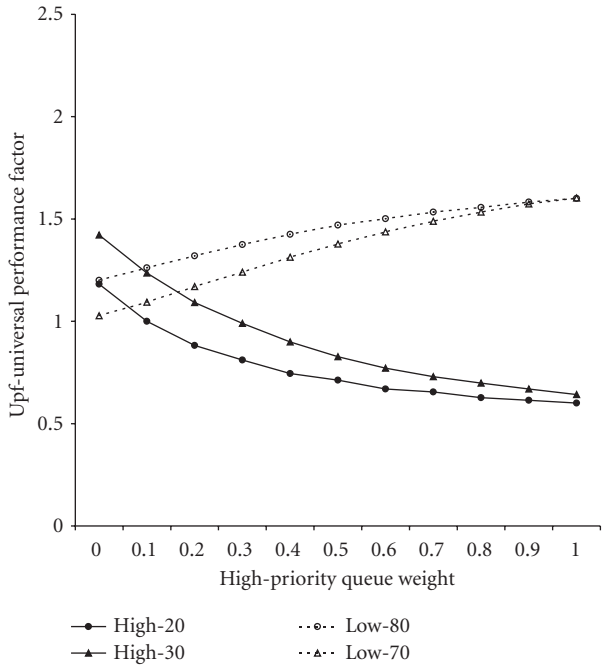

Figure 11: Universal performance factor for different priority classes under varying high-priority queue weights and high load.

case of throughput (Figure 9), the decreased network load results in small delays for packets for "reasonable" settings of the high-priority queue weight, and, therefore, the margins for improvement/deterioration are small. For the last portion of the curves (high-priority queue weight between 0.7 and 1), both the gains of high-priority packets and the losses for low-priority ones are less than 3% in all cases.

Finally, Figure 11 depicts the *universal performance factor* (Upf) for different priority classes under varying high-priority queue weights, high network load, and two high/low packet ratios (20/80 and 30/70). Similarly to the full load case, since the individual performance factors (throughput and delay) combined in Upf evolve along a specific pattern (i.e., high-priority packets are served better as the high-priority queue weight increases while the inverse holds for low-priority packets), the same pattern is exhibited by the Upf too: its value drops (i.e., improves) for high-priority packets as the high-priority queue weight increases, while for low-priority packets, its value rises (i.e., deteriorates) as the high-priority queue weight increases. Note that the absolute values of Upf in Figure 11 are higher (i.e., worse) than the respective values of the full-load case (Figure 8), indicating that network resources are underutilized.

6. Conclusions

In this paper, we have addressed the performance evaluation of a dual-priority, single-buffered, 6-stage MIN, employing the class-based weighted fair queuing packet scheduling algorithm. We have presented analytical equations for modelling their operation, employing a scheme that takes into account both the previous and the last state of the SEs' queues, providing thus better accuracy than schemes considering only the last state.

We have also evaluated through simulations the overall performance of the MIN and the quality of service offered to each priority class under varying high-priority queue weights, different high-/low-priority packet ratios (20/80 and 30/70), and different MIN loads (full load and high load) when using the class-based weighted fair queuing algorithm and compared these results against the strict priority algorithm. The performance evaluation results show that the strict priority algorithm does offer the high-priority packets better quality of service, but on the other, hand it degrades the overall MIN performance and significantly degrades the quality of service offered to low-priority packets. Configuring the high-priority queue weight in the range [0.7, 1] has marginal effects both on the overall MIN performance and the QoS offered to packets of different priority classes. On the other hand, setting the high-priority queue weight in the range [0.45, 0.7) appears to achieve a good balance among overall MIN performance, prioritization of high-priority packets, and acceptable QoS for low-priority packets (always considering the high-/low-priority packet ratios 20/80 and 30/70). MIN designers and operators can use the results presented in this paper to optimally configure the weights of the queues, taking into account the QoS they want to offer to packets of different priorities and the overall MIN performance they want to achieve.

Future work will focus on examining other load configurations, including hot-spot and burst loads, as well as different buffer sizes and handling schemes.

References

[1] B. Prabhakar and N. McKeown, "On the speedup required for combined input- and output-queued switching," *Automatica*, vol. 35, no. 12, pp. 1909–1920, 1999.

[2] X. Li, L. Mhamdi, J. Liu, K. Pun, and M. Hamdi, "Architectures of internet switches and routers," in *High-performance Packet Switching Architectures*, I. Elhanany and M. Hamdi, Eds., Springer, London, UK, 2007.

[3] A. Elwalid and D. Mitra, "Analysis, approximations and admission control of a multi-servicemultiplexing system with priorities," in *14th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '95)*, pp. 463–472, April 1995.

[4] J. Y. Kim, T. Z. Shin, and M. K. Yang, "Analytical modeling of a multistage interconnection network with buffered axa switches under hot-spot environment," in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '07)*, pp. 137–140, August 2007.

[5] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, 1996.

[6] C. Guo, "SRR: an O(1) time-complexity packet scheduler for flows in multiservice packet networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 1144–1155, 2004.

[7] A. Elwalid and D. Mitra, "Design of generalized processor sharing schedulers which statistically multiplex heterogeneous QoS classes," in *18th Annual Joint Conference of the IEEE Computer and Communications Societie (INFOCOM '99)*, vol. 3, pp. 1220–1230, March 1999.

[8] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Journal of Internetworking Research and Experience*, vol. 1, no. 1, pp. 3–26, 1990.

[9] J. F. Shortle and M. J. Fischer, "Approximation for a two-class weighted fair queueing discipline," *Performance Evaluation*, vol. 67, no. 10, pp. 946–958, 2010.

[10] L. Zhang, "VirtualClock: a new traffic control algorithm for packet-switched networks," *ACM Transactions on Computer Systems*, vol. 9, no. 2, pp. 101–124, 1991.

[11] S. Jamaloddin Golestani, "Self-clocked fair queueing scheme for broadband applications," in *Annual Joint Conference of the IEEE Computer and Communications Societie (INFOCOM '94)*, vol. 2, pp. 636–646, June 1994.

[12] D. C. Vasiliadis, G. E. Rizos, C. Vassilakis, and E. Glavas, "Modelling and performance evaluation of a novel internal priority routing scheme for finite-buffered multistage interconnection networks," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 26, no. 5, pp. 381–397, 2011.

[13] M. Moudi and M. Othman, "A challenge for routing algorithms in optical multistage interconnection networks," *Journal of Computer Science*, vol. 7, no. 11, pp. 1685–1690, 2011.

[14] M. Othman and T. D. Shahida, "The development of crosstalk-free scheduling algorithms for routing in optical multistage interconnection networks," Trends Telecommunication Technologies, March 2010, http://www.intechopen.com/download/pdf/pdfs_id/9696.

[15] Cisco Systems., "QoS Scheduling and Queueing on the Catalyst 3550 Switches," July 2011, http://www.cisco.com/en/US/tech/tk389/tk813/technologies_tech_note09186a00801558cb.shtml.

[16] Hewlet Packard, "3Com SuperStack 3 Switch 3800—Overview," July 2011, http://bizsupport1.austin.hp.com/bizsupport/TechSupport/Document.jsp?objectID=c02642521&printver=true.

[17] Nortel Networks, "Nortel Ethernet Switch 460/470 Overview—System Configuration," July 2011, http://support.avaya.com/css/P8/documents/100099692.

[18] Avaya Inc, "Automatic QoS Technical Configuration Guide for the ERS 4500, 5000, Avaya BCM 50, 450, Avaya CS 1000, Avaya CS 2100 and Avaya SRG 50," July 2011, http://support.avaya.com/css/P8/documents/100123842.

[19] Dax networks, "Dax Dx-5048GM technical specifications," http://www.daxnetworks.com/products-dec2010/switches/switches/dax%20dx-5048gm.asp?Page=3&Print=.

[20] Cisco Systems, "Class-Based Weighted Fair Queueing," Chapter in Cisco IOS Software Releases 12.0 T, 2010, http://www.cisco.com/en/US/docs/ios/12_0t/12_0t5/feature/guide/cbwfq.html.

[21] Hewlet-Packard, "HP ProCurve Secure Router 7000dl Series," July 2011, http://www.hp.com/rnd/pdfs/datasheets/ProCurve_Secure_Router_7000dl_Series.pdf.

[22] G. A. Abandah and E. S. Davidson, "Modeling the communication performance of the IBM SP2," in *10th International Parallel Processing Symposium*, pp. 249–257, April 1996.

[23] C.-H. Choi and S.-C. Kim, "Hierarchical multistage interconnection network for sharedmemory multiprocessor system," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 468–472, 1997.

[24] J. Torrellas and Z. Zhang, "The performance of the cedar multistage switching network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 4, pp. 321–336, 1997.

[25] R. Y. Awdeh and H. T. Mouftah, "Survey of ATM switch architectures," *Computer Networks and ISDN Systems*, vol. 27, no. 12, pp. 1567–1613, 1995.

[26] T. Soumiya, K. Nakamichi, S. Kakuma, T. Hatano, and A. Hakata, "The large capacity ATM backbone switch 'FETEX-150 ESP'," *Computer Networks*, vol. 31, no. 6, pp. 603–615, 1999.

[27] E. S. H. Tse, "Switch fabric architecture analysis for a scalable bi-directionally reconfigurable IP router," *Journal of Systems Architecture*, vol. 50, no. 1, pp. 35–60, 2004.

[28] G. F. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *1st Annual Symposium on Computer Architecture*, pp. 21–28, 1973.

[29] D. C. Vasiliadis, G. E. Rizos, C. Vassilakis, and E. Glavas, "Performance evaluation of two-priority network schema for single-buffered delta networks," in *18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, September 2007.

[30] D. C. Vasiliadis, G. E. Rizos, and C. Vassilakis, "Improving performance of finite-buffered blocking delta networks with 2-class priority routing through asymmetric-sized buffer queues," in *4th Advanced International Conference on Telecommunications (AICT '08)*, pp. 23–29, June 2008.

[31] S. Kumar, "Mathematical modelling and simulation of a buffered Fault Tolerant Double Tree Network," in *15th International Conference on Advanced Computing and Communications (ADCOM '07)*, pp. 422–431, December 2007.

[32] C. Bouras, J. Garofalakis, P. Spirakis, and V. Triantafillou, "An analytical performance model for multistage interconnection networks with finite, infinite and zero length buffers," *Performance Evaluation*, vol. 34, no. 3, pp. 169–182, 1998.

[33] T. Lin and L. Kleinrock, "Performance analysis of finite-buffered multistage interconnection networks with a general traffic pattern," in *International Conference on Measurement and Modeling of Computer Systems*, pp. 68–78, San Diego, Calif, USA, 1991.

[34] D. Tutsch and G. Hommel, "Comparing switch and buffer sizes of multistage interconnection networks in case of multicast traffic," in *High Performance Computing Symposium (HPC '02)*, pp. 300–305, San Diego, Calif, USA, 2002.

[35] A. K. Gupta, L. O. Barbosa, and N. D. Georganas, "Switching modules for ATM switching systems and their interconnection networks," *Computer Networks and ISDN Systems*, vol. 26, no. 4, pp. 433–445, 1993.

[36] D. C. Vasiliadis, G. E. Rizos, and C. Vassilakis, "Routing and performance evaluation of dual priority Delta networks under hotspot environment," in *1st International Conference on Advances in Future Internet (AFIN '09)*, pp. 24–30, June 2009.

[37] D. Vasiliadis, G. Rizos, and C. Vassilakis, "Performance study of multilayered multistage interconnection networks under hotspot traffic conditions," *Journal of Computer Systems, Networks, and Communications*, vol. 2010, Article ID 403056, 11 pages, 2010.

[38] 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*, vol. 15, no. 2, pp. 131–155, 2006.

[39] J. Garofalakis and E. Stergiou, "Performance evaluation for multistage interconnection networks servicing unicast and multicast traffic (by Partial Operation)," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS '09)*, pp. 311–318, July 2009.

[40] D. Tutsch and G. Hommel, "Multilayer multistage interconnection networks," in *Design, Analysis, and Simulation of*

Distributed Systems (DASD'03), pp. 155–162, Orlando, Fla, USA, 2003.

[41] M. Nabeshima, "Packet-based scheduling for ATM networks based on comparing a packet-based queue and a virtual queue," *IEICE Transactions on Communications*, vol. E82-B, no. 6, pp. 958–961, 1999.

[42] Y. Mun and H. Y. Youn, "Performance analysis of finite buffered multistage interconnection networks," *IEEE Transactions on Computers*, vol. 43, no. 2, pp. 153–162, 1994.

[43] J. H. Patel, "Processor-memory interconnections for mutliprocessors," in *6th Annual Symposium on Computer Architecture*, pp. 168–177, New York, NY, USA, 1979.

[44] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Transactions on Computers*, vol. 24, no. 12, pp. 1145–1155, 1975.

[45] G. B. Adams and H. J. Siegel, "The extra stage cube: a fault-tolerant interconnection network for supersystems," *IEEE Transactions on Computers*, vol. 31, no. 5, pp. 443–454, 1982.

[46] W. R. Stevens, *TCP/IP Illustrated. Volume 1. The Protocols*, Addison-Wesley, 10th edition, 1997.

[47] C. A. Waldspurger and W. E. Weihl, "Lottery scheduling: flexible proportional-share resource management," in *Proceedings of the Symposim on Operating System Design and Implementation*, November 1994.

[48] J. Liebeherr and E. Yilmaz, "Workconserving vs. non-workconserving packet scheduling," in *17th International Workshop on Quality of Service (IWQoS '99)*, pp. 248–256, 1999.

[49] 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, 1991.

[50] D. C. Vasiliadis, G. E. Rizos, and C. Vassilakis, "Performance analysis of blocking Banyan swithces," in *Proceedings of (CISSE '06)*, December 2006.

[51] Y. C. Jenq, "Performance analysis of a packet switch based on single-buffered banyan network," *IEEE Journal on Selected Areas in Communications*, vol. 1, no. 6, pp. 1014–1021, 1983.

[52] D. C. Vasiliadis, G. E. Rizos, C. Vassilakis, and E. Glavas, "Routing and performance analysis of double-buffered omega networks supporting multi-class priority traffic," in *3rd International Conference on Systems and Networks Communications (ICSNC '08)*, pp. 56–63, October 2008.