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

D. C. Vasiliadis[a,b], G. E. Rizos[a,b], C. Vassilakis[a]
[a]Department of Computer Science and Technology
University of Peloponnese
Tripolis, Greece
[b]Technological Educational Institute of Epirus,
Arta, Greece
dvas@uop.gr, georizos@uop.gr, costas@uop.gr

***Abstract*— Contemporary networks support multiple priorities, aiming to differentiate the QoS levels offered to individual traffic classes. Support for multiple priorities necessitates the introduction of a scheduling algorithm, 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 starvation, i.e. receiving no service during periods of high-priority traffic. CBWFQ effectively thus offers low-priority queues the opportunity to transmit packets even though the high-priority queues are not empty. In this paper, we present the modeling and performance evaluation of a single-buffered, four-priority multistage interconnection network (MIN) operating under the CBWFQ scheduling policy. Performance evaluation is conducted through simulation, and the performance metrics obtained can be used by MIN designers to set the appropriate queue weights according to the expected traffic and the desired QoS levels for each priority class, delivering efficient thus systems.**

Keywords-Multistage Interconnection Networks; Delta Netwoks; Performance Evaluation

I. INTRODUCTION

During the last two decades, technology has offered the potential for a dramatic increase in network speeds, whiles the explosion of content availability and the introduction of new services, such as streaming media and file sharing, has multiplied the amount of network traffic. In contemporary networks, not all packets are treated equally: different priorities are assigned to packets entering the networks, and subsequently the network employs scheduling algorithms which take into account packet priorities in the process of selecting each time the next packet to transmit over the data link, offering thus different quality-of-service (QoS) levels to packets of different priority classes.

In this context, a number of packet scheduling algorithms have been proposed and used in networks, with the most widespread ones being strict priority queuing [43], round-robin [63] and its variations (e.g. weighted round-robin [17] [31], deficit round-robin [46], smoothed round-robin [22]), generalized processor sharing (GPS) [18], weighted fair queuing (P-GPS) [15], class-based weighted fair queuing (CBWFQ) [45], virtual clock [64] and self-clocked fair queuing [21].

The selection of the packet scheduling algorithm can radically affect both the QoS offered to packets transmitted through the network and the overall performance of the network. This is due to the fact that different algorithms aim to optimize different aspects of packet QoS and network behavior, such as throughput, delay, delay jitter, prioritization and fairness. Besides these high-level goals, when choosing the packet scheduling algorithm that will be used in a network, implementation-level characteristics of candidate algorithms are also taken into account: for instance, [22] reports that two important factors are the algorithms' space and time complexity (since they affect the amount of memory and the processing power required to implement the algorithm, respectively) and the ease of implementation (since more complex algorithms are generally more demanding in space and time, while their implementations are more error-prone).

In the commercial product domain, among the algorithms listed above, *strict priority queuing* (i.e. first servicing high priority packets and examining lower priority ones only when higher priority ones are not waiting to be serviced), *weighted round robin* (i.e. dividing the available bandwidth into –possibly unequal- portions and assigning one portion to each priority queue) and *class-based weighted fair queuing* [i.e. in the presence of N currently active data flows with respective weights $w_1,w_2...w_N$, data flow i will achieve an average data rate of $R*w_i/(w_1+w_2+...+w_N)$, where R is the data link rate] [45] have been adopted by the industry and implemented in most commercial products (e.g. [7], [25], [41], [4], [14], [8], [26]). These algorithms are preferred since they exhibit the following desirable characteristics (a) ease of implementation and verification (b) good exploitation of the available network bandwidth (c) limited processing power and memory requirements and (d)

network administrators find them easy to understand and configure.

Regarding the communication infrastructure internal architecture, multistage interconnection networks (MINs) with crossbar switching elements (SEs) are frequently used for implementing the interconnection between processors and memory modules in parallel multiprocessor systems [1], [13], [51], and are also considered to be a very efficient means for implementing network communication devices such as gigabit Ethernet switches, terabit routers and ATM switches [5], [47], [52]. MIN technology offers the significant advantages of a low cost/performance ratio and has the potential to route multiple communication tasks concurrently, resulting in good exploitation of the available hardware. MINs with the Banyan [20] property are generally preferred against non-Banyan MINs, since the latter have found to be more expensive than Banyan networks and more complex to control.

Insofar, the performance of multi-priority MINs operating under the strict priority queuing algorithm has been studied extensively, through both analytical methods and simulation experiments (e.g. [56], [57], [58], [32], [6], [35], [53]), considering various buffer sizes (mainly buffer lengths 1, 2 and 4), schemes for allocating available buffer space to different priority classes (symmetric vs. asymmetric [57]), arrival processes (e.g. uniform vs. bursty [23]), traffic patterns (e.g. uniform vs. hotspot [59],[60],[31]; unicast vs. multicast [24],[48]) and internal MIN architectures (e.g. single-layer vs. multi-layer [54]). These studies have shown that when network load increases (and more specifically, when the packet arrival probability λ increases beyond 0.6), the QoS offered to low priority packets sharply drops, with throughput significantly deteriorating and delay sharply rising.

In order to rectify this situation, class-based weighted fair queuing (CBWFQ) can be used as a packet scheduling algorithm instead of strict priority queuing; this stems from the fact that one of the design goals of CBWFQ is to increase fairness, giving lower-priority queues the opportunity to transmit packets even in cases that higher-priority queues are not empty. As compared to using weighted round-robin, CBWFQ has the advantages of being able to guarantee fair link sharing, while it doesn't pose the requirement of knowing the mean packet size of each connection in advance [38]. Insofar, however, no studies have been conducted to quantify (a) the gains obtained for low-priority packets (and conversely the losses incurred for high-priority packets) by the introduction of the CBWFQ packet scheduling algorithm and (b) how queue weight assignment affects the overall performance of the MIN network and the QoS offered to individual priority classes. Note that the performance of CBWFQ has been studied for other network classes, notably torus-based networks and mesh networks, the results obtained from these studies cannot be directly used for the case of MINS: indeed, in both torus and mesh networks, multiple paths exist between network elements [12] (contrary to Banyan MINs considered in this paper), and hence routing algorithms are adapted to exploit this feature for both performance and fault tolerance. Additionally, many studies regarding mesh networks have been made in the context of wireless networks including ad-hoc ones [3], in which cases (a) routers typically are also destination nodes themselves and (b) the topology of the network constantly changes, due to mobility.

Taking the above facts into account, in this paper, we present a simulation-based performance evaluation for single-buffered MINs natively supporting four priority classes and employing the CBWFQ packet scheduling algorithm. In this performance evaluation, we calculate the QoS offered to packets of different priority classes, focusing in the areas of high network loads (in which, under the strict priority algorithm, the QoS offered to lower-priority packets deteriorates) and under different ratios of packets in the distinct priority classes. We also study the effect of queue weight assignment in the QoS offered to packets of different priorities. The performance metrics obtained can be used by MIN designers to set the appropriate queue weights according to the expected traffic and the desired QoS levels for each priority class, delivering efficient thus systems.

While the 802.1D standard [28] specifies eight priority levels and the Diffserv standard [40] specifies six "class selectors", it has been anticipated that few switches will actually provide support for eight priority classes [19], and hence IEEE 802.1Q provides recommended mappings from the eight priority classes specified in 802.1D to fewer queues [29]. Many contemporary switches prioritize packets through a process involving the steps of *classification*, *marking* and *queuing* (with a *policing* step also appearing in some cases) [39][9], and the outcome of this process is the placement of the packets in a maximum of four queues (e.g. [27][16][10]), with eight queues being supported only by few high-end switches (e.g. [9], [11]). Thus, in this paper we focus on studying MINs that natively support four priority levels.

The rest of this paper is organized as follows: in section II we present the quad-priority MIN and give details on its operation and the class-based weighted fair queuing packet scheduling algorithm. In sections III and IIV we present the performance metrics and the simulation results, respectively, while in section V conclusions are drawn and future work is outlined.

II. QUAD-PRIORITY MIN AND THE CLASS-BASED WEIGHTED FAIR QUEUING SCHEDULING ALGORITHM

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. Its main characteristics are its topology, routing algorithm, switching strategy and flow control mechanism.

All types of blocking Multistage Interconnection Networks (Delta Networks [42], Omega Networks [33] and Generalized Cube Networks [2]) with the Banyan property which is defined in [20] are characterized by the fact that there is exactly one path from each source (input) to each sink (output). Banyan MINs are multistage self-routing switching fabrics. Consequently, each SE of k^{th} stage, where k=1...n can decide in which output port to route a packet, depending only on the corresponding k^{th} bit of the destination address.

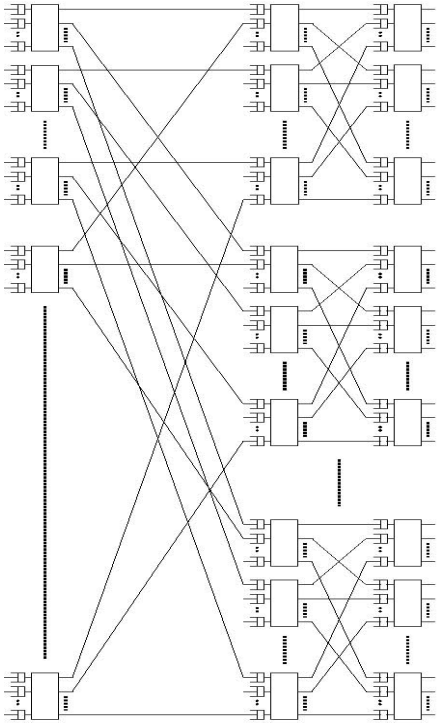

Figure 1. A 3-stage Delta Network

An (N X N) MIN can be constructed by $n=\log_c N$ stages of (cxc) SEs, where c is the degree of the SEs. At each stage there are exactly N/c SEs. Consequently, the total number of SEs of a MIN is $(N/c)*\log_c N$. Thus, there are $O(N*\log N)$ interconnections among all stages, as opposed to the crossbar network which requires $O(N^2)$ links. A typical configuration of a (N X N) Delta Network is depicted in figure 1. Regarding priority handling, each SE is modelled by as an array of p non-shared buffer queue pairs, where p is the number of distinct priority classes supported by the network, with the i^{th} element of the array being dedicated to packets of priority class i. Within each pair, one buffer queue is dedicated for the upper queuing bank and the other for the lower bank. In this paper, we consider a quad-priority Delta Network that operates under the following assumptions:

- The network clock cycle consists of two phases. In each time slot two phases take place. In the first phase, control information passes via the network from the last stage to the first one. In the second phase, packets flow from the first stage towards the last, in accordance to the flow control information.
- At each input of every switch of the MIN only one packet can be accepted within a time slot which is marked by a priority tag, and it is routed to the appropriate class queue (figure 2). The domain value for this special priority tag in the header field of the packet determines its i-class priority, where i=1…p. Notably, provisions for packet priorities can be found in early protocol specifications, such as the case of TCP out-of-band/expedited data, which are normally prioritized against normal connection data [49], while more recent specifications such as 802.1D [28] and Diffserv [40] have increased the number of available priority classes.

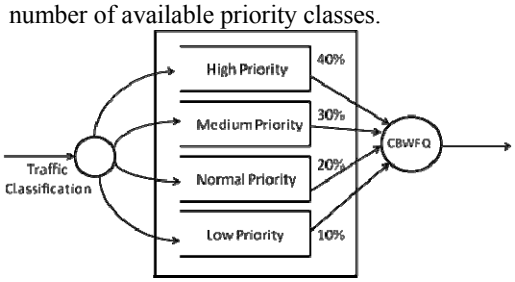

Figure 2. Class-based weighted fair queuing algorithm

- The arrival process of each input of the network is a simple Bernoulli process, i.e. the probability that 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_i probabilities, which one represents the arrival probability for i-priority packets, where i=1...p. It holds that $\lambda = \sum_{i=1}^{p} \lambda_i$
- An i-class priority packet arriving at the first stage is discarded if the corresponding i-class priority buffer of the SE is full, where i=1...p. This is indicated through a control signal to the source of the packet (typically, network protocol software running in a host or a network apparatus connected to the input port), and the source will arrange for the packet retransmission, according to the rules of the employed protocol. The responsibility for handling cases that such discardings/retransmissions result into expiration of the packet's TTL (and therefore its elimination from the network) is assumed by higher-layer network protocols.
- A backpressure blocking mechanism is used, according to which an i-class priority packet is blocked at a stage if the destination of the

```
/* Simulation initialization.
      entry: qw[4] --> queue weights
      exit: probabilities[16][4] --> probabilities for all possible queue
      state combinations. Index is a 4-bit quantity with each bit being
      1 if the corresponding queue is full and 0 if the corresponding
      queue is empty (its bits are read right to left). For example:
      index = 3  --> 0011 binary --> queues 0 and 1 have packets
      index = 13 --> 1101 binary --> queues 0, 2 and 3 have packets
      index = 15 --> 1111 binary --> all queues have packets
      index 0 (no queue has packets) is unused

e.g. qw[] = {40%, 30%, 20%, 10%}

probabilities[1] = {1, -1, -1, -1} -- only 1 queue has packets, gets 100%
probabilities[5] = {2/3, -1, 1, -1} -- 3rd element is 1 because
                probabilities are cumulative
probabilities[13] = {4/7, -1, 6/7, 1}
probabilities[15] = {4/10, 7/10, 9/10, 1}
*/

for (i = 1; i < 15; i++) {
      /* find total weight of non-empty queues */
      totWeight = 0;
      index = i;
      for (j = 0; j < 4; j++) {
            if (index & 1 == 1) { /* test rightmost bit */
                  totWeight += qw[j];
                  index = index >> 1; /* shift bits right */
            }
      }
      /* find proportions and compute cumulative distribution */
      totalProbs = 0;
      lastNonEmpty = 0;
      index = i;
      for (j = 0; j < 4; j++) {
            if (index & 1 == 1) { /* test rightmost bit */
                  lastNonEmpty = j;
                  totalProbs += qw[j] / totalWeight;
                  probabilities[i][j] = totalProbs;
            }
            else
                  probabilities[i][j] = -1;
            index = index >> 1; /* shift bits right */
      }
      /* set the cumulative probability of the last queue to 1 to fix
            arithmetic rounding errors */
      probabilities[i][lastNonEmpty] = 1;
}

/* end initialize */
```

Figure 3. Initialization of the class-based weighted fair queuing algorithm

corresponding *i*-class priority buffer at the next stage is full, w*h*ere *i*=1...*p*.

- All *i*-class priority packets are uniformly distributed across all the destinations and each *i*-class priority queue uses a FIFO policy for all output ports, where *i*=1...*p*.
- 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.
- Regarding the strict priority scheduling algorithm the lower-priority queues are only serviced if the higher-priority queues contains no packets. On the other hand, 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 $p(q_i) = \frac{w_i}{\sum_{j \in S} w_j}$, where w_k is the weight assigned to queue *k* [45]. This is analogous to lottery scheduling used in operating systems [62]. We note here that the class-based weighted fair queuing algorithm considered in this paper is *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 [34]. 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.

  Figures 3 and 4 illustrate the operation of the class-based weighted fair queuing algorithm as implemented in the simulation, with the code in Figure 3 being the initialization, and the code in Figure 4 being executed each time a packet is forwarded. The initialization step includes the computation of cumulative probability distributions and their storage into arrays, in order to speed up the packet selection step that is repeatedly executed.
- 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

```
/* Simulation cycle. Select the packet to forward */

/* First, determine non-empty queues & select the appropriate index */
index = 0;
for (i = 3; i >= 0; i--) {
      index = index << 1; /* shift bits left */
      if (! isEmpty(queue[i]))
            index = index | 1; /* set rightmost bit */
}
if (index > 0) { /* at least one queue has packets, select queue and
forward packet */
      packetProb = random(); /* 0 <= packetProb < 1 */
      for (i = 0; i < 4; i++)
            if (probabilities[index][i] > packetProb)
                  break;
      }
      /* now i holds the queue to transmit from */
      forward_packet_from_queue(i);

}
```

Figure 4. Packet selection and forwarding in the the class-based weighted fair queuing algorithm

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.

- All SEs have deterministic service time.
- 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.

III. Performance Evaluation Metrics for Quad-Priority MINs

In this section the two most important network performance factors, namely *packet throughput* and *delay* are analyzed and modelled for the case of Quad-Priority MINs. The *universal performance factor* introduced in [55], which combines the above two metrics into a single one is also considered.

In order to evaluate the performance of multi-priority (*N*X*N*) MIN the following metrics are used. Let Th_{avg} and D_{avg} be the *average throughput* (*bandwidth*) and *average delay* of a MIN respectively.

Normalized throughput Th [30] is the ratio of the *average throughput* Th_{avg} to number of network outputs *N*. Formally, *Th* can be expressed by

$$Th = \frac{Th_{avg}}{N} \quad (1)$$

and reflects how effectively network capacity is used.

Relative normalized throughput RTh(i) of i-class priority traffic, where i=1..p is the *normalized throughput Th(i)* of *i*-class priority packets divided by the corresponding-class *offered load* $\lambda(i)$ of such packets.

$$RTh(i) = \frac{Th(i)}{\lambda(i)} \quad (2)$$

This extra normalization of each class-priority traffic leads to a common value domain needed for comparing their absolute performance values in all configuration setups.

Normalized packet delay D(i) of i-class priority traffic, where i=1..p is the ratio of the $D_{avg}(i)$ to the minimum packet delay which is simply the transmission delay $n*nc$ (i.e. zero queuing delay), where $n=\log_2 N$ is the number of intermediate stages and *nc* is the network cycle. Formally, $D(i)$ can be defined as

$$D(i) = \frac{D_{avg}(i)}{n * nc} \quad (3)$$

The definition of normalized delay $D(i)$ effectively extends the definition of normalized delay in [30] to consider the different priority classes.

Universal performance factor Upf(i) of i-class priority traffic, where i=1..p is defined by a relation involving the two major above normalized factors, $D(i)$ and $Th(i)$*:* the performance of a MIN is considered optimal when $D(i)$ is minimized and $Th(i)$ is maximized, thus the formula for computing the *universal performance factor* arranges so that the overall performance metric follows that rule. Formally, $Upf(i)$ can be expressed by

$$Upf(i) = \sqrt{w_d * D(i)^2 + w_{th} * \frac{1}{Th(i)^2}} \quad (4)$$

where w_d and w_{th} denote the corresponding *weights* for each factor participating in the *U,* 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, we normalize 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 (4) can be replaced by:

$$Upf(i) = \sqrt{w_d * \left(\frac{D(i) - D(i)^{\min}}{D(i)^{\min}}\right)^2 + w_{th} * \left(\frac{RTh(i)^{\max} - RTh(i)}{RTh(i)}\right)^2} \quad (5)$$

where $D(i)^{min}$ is the minimum value of *normalized packet delay* $D(i)$ and $RTh(i)^{max}$ is the maximum value of *Relative normalized throughput* $RTh(i)$. Consistently to equation (4), when the *universal performance* factor $Upf(i)$, as computed by equation (5) is close to 0, the performance a MIN is considered optimal whereas, when the value of $Upf(i)$ increases, its performance deteriorates. Finally, taking into account that the values of both *delay* and *throughput* appearing in equation (5) are normalized, $D(i)^{\min} = RTh(i)^{\max} = 1$, thus the equation can be simplified to:

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

The definition of universal performance $Upf(i)$ effectively extends the definition of universal performance factor in [55] to consider the different priority classes.

In this study, when calculating the value of the above combined factor *Upf*, we have considered the individual performance factors (*packet throughput* and *delay*) to be of equal importance ($w_d = w_{th} = 1$). This is not necessarily true for all application classes, e.g. for batch data transfers *throughput* is more important, whereas for streaming media the *delay* must be optimized.

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

Buffer-size b(i) *of an i-class priority queue*, where i=1..p is the maximum number of such packets that the corresponding i-class input buffer of a SE can hold. In this paper we consider symmetric-sized single-buffered $b(i)$=1 MINs, where i=1..4. It is worth noting that a buffer size of $b(i)$=1 is being considered since under this setting each SE egress link is effectively equipped with four buffer positions (one buffer space for each distinct priority). Studies on single-priority architectures have shown that increasing the buffer size beyond four would lead to excessive delays [55], hence in this study we have fixed the *per priority* buffer size to 1.

Offered load λ(i) *of i-class priority traffic,* where i=1..p is the steady-state fixed probability of such arriving packets at each queue on inputs. It holds that $\lambda = \sum_{i=1}^{p} \lambda(i)$, where λ represents the total arrival probability of all packets. In our simulation λ is assumed to be λ = 0.1, 0.2… 0.9, 1.

Ratio of i-class priority offered load r(i), where i=1..p expressed by $r(i)=\lambda(i)/\lambda$. It is obvious that $\sum_{i=1}^{p} r(i) = 1$. In this paper the ratios of high, medium, normal and low priority packets are assumed to be $r(4)$=0.20, $r(3)$=0.25, $r(2)$=0.25 and $r(1)$=0.30 respectively.

Weight of i-class priority queues w(i) is the percentage rate of processor dedicated to i-class priority packets in each queue by the applied scheduling algorithm. In the case of CBWFQ discipline the *weight* of a class-priority expresses the probability that a particular class queue is examined first; this probabilistic mechanism applied individually in each SE for every cycle repeatedly. In this paper we consider two different case studies. In the first scenario the *weights* of higher priority-classes are considered to be greater than those of lower ones [$w(4)$=0.40, $w(3)$=0.30, $w(2)$=0.20 and $w(1)$=0.10], while at the second configuration all priority classes are assumed to have equal weights $w[i]$=0.25, where i=1..4.

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

IV. Simulation and Performance Results

In this paper we developed a special simulator in C++, capable of handling quad-priority MINs using the class-based weighted fair queuing. Each (2X2) SE was modeled by eight non-shared buffer queues, where buffer operation was based on the first come first serviced principle; each egress link of the SE (upper and lower) is provided with four buffer queues, corresponding to the four priority levels.

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 each class-priority offered load,* the *weight of each class-priority queue,* 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.

A. Simulator validation

Since no other related works on simulators for multi-priority MINs operating under class-based weighted fair queuing scheduling discipline have been reported insofar in the literature, we validated our simulator only against those that use strict priority scheduling. In the case of single-priority traffic p=1, we noticed that all simulation experiments were in close agreement with the results reported in [56] (fig. 2 in [56]), and -notably- with Theimer's model [50], which is considered to be the most accurate one in comparison with the other two classical models [37],[30]. In dual-priority MINs (p=2) we compared our measurements against those obtained from Shabtai's Model reported in [44], and have found that both results are in close agreement (maximum difference was only 3.8%).

B. 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* [$th=th(h)+th(m)+th(n)+th(l)$] of a MIN using a quad-priority scheme vs. the offered load, for different queue weight assignments. In Figure 5, curve PQ corresponds to the *total normalized throughput* of a 6-stage MIN operating under the strict priority queue scheduling algorithm, while curves CBWFQ[H,M,N,L] indicate the *total normalized throughput* of the same MIN operating under the CBWFQ scheduling policy and having the weights of its high, medium, normal and low packets set to H, M, N and L, respectively. In all cases, the ratio of high, medium, normal and low packets against the overall network load is set to 20%, 25%, 25% and 30% correspondingly. Under this load mixture, the queue weight setting 25/25/25/25 roughly corresponds to a setup with no priorities (the correspondence is not exact because the load ratios for different priority classes are not equal), while queue weight setting 40/30/20/10 corresponds to a "mild prioritization"

scheme where e.g. a high-priority packet has approximately double probability to be transmitted as compared to a medium-priority packet, when they contend for the same output link (high-priority packets have a 40% ratio of the bandwidth being the 20% of the overall traffic, while medium-priority packets have a 30% ratio of the bandwidth being the 25% of the overall traffic).

We can notice here that by employing the CBWFQ algorithm, the overall MIN throughput increases, as compared to the PQ algorithm with the increment ranging from 0.9% (λ=0.6) to 2% (λ=1). This can be attributed to the fact that under CBWFQ, network resources are better exploited; this particularly applies to network buffers dedicated to lower-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 new lower-priority packets arrive at the corresponding SE. Nevertheless, the primary goal of classifying the packets into four priority classes is to provide better QoS to higher priority ones. This goal can be also achieved under the CBWFQ algorithm, by setting the weight of the higher-priority queues to a value greater than the anticipated load of packets with the corresponding priorities. 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 higher-priority packets.

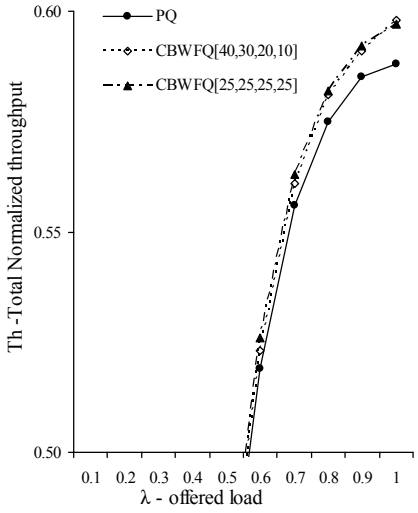

Figure 5. Overall MIN throughput under varying scheduling algorithms and queue weight settings

In Figure 5 we can also notice that the overall MIN performance is practically not affected at all when shifting from the "no priorities" scheme (queue weight setting: 25/25/25/25) to the "mild prioritization" scheme (queue weight setting: 40/30/20/10).

In the following paragraphs, we discuss the QoS level delivered to packets of different priority classes under the above queue weight settings when the CBWFQ algorithm is employed.

C. *Quad-Priority MINs Performance under High Network Load*

In this subsection we examine the effect of queue weight setting on the QoS offered to packets of different priority classes under the CBWFQ algorithm, and we also compare these QoS levels to the corresponding ones delivered by the strict priority queuing scheduling algorithm,

Figures 6, 7, 8 and 9 depict the relative normalized throughput for packets belonging to the high, medium, normal and low priority classes, respectively. In these figures (and subsequent ones also), curves PQ correspond to the *total normalized throughput* of a 6-stage MIN operating under the strict priority queue scheduling algorithm, while curves CBWFQ[25,25,25,25] and CBWFQ[40,30,20,10] correspond to the "no priority" and "mild prioritization" scenarios described above. In all cases, the ratio of high, medium, normal and low packets against the overall network load is set to 20%, 25%, 25% and 30% respectively.

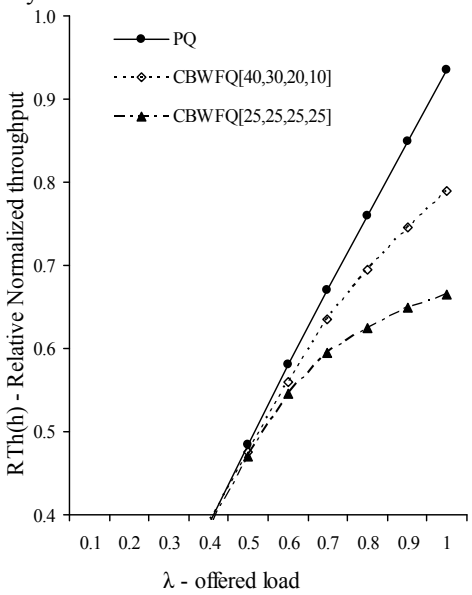

Figure 6. Relative normalized throughput for high-priority packets vs. offered load

Figure 6 shows that, while under strict priority queuing the normalized throughput for high priority packets is close to optimal, under CBWFQ the respective normalized throughput (expectedly) drops. The deterioration ranges from 5.2% (λ=0.7) to 15.5% (λ=1) for the "mild prioritization" scenario, while in the "no priorities" scenario the corresponding drop ranges from 11.2% (λ=0.7) to 28.8% (λ=1). Similar observations hold for medium priority

packets (Figure 7), which under the "mild prioritization" scenario exhibit a drop in normalized throughput varying from 6.8% (λ=0.7) to 16.9% (λ=1) and a drop in the same metric ranging from 12.4% (λ=0.7) to 27.5% (λ=1) under the "no priorities" scenario.

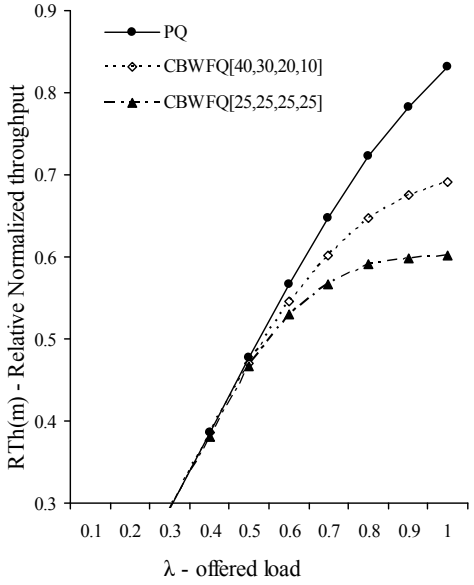

Figure 7. Relative normalized throughput for medium-priority packets vs. offered load

Figures 8 and 9 show that using the CBWFQ algorithm instead of the strict priority queuing one is beneficial for normal- and low-priority packets (especially for the latter). Indeed, under the "mild prioritization" scenario, the gains for normal- and low-priority packets scale up to 11.0% and 101.7% respectively (in both cases for offered load λ=1), while under the "no priorities" scenario, the corresponding gains are 11.0% and 181.3%, again for λ=1. Interestingly, the *relative normalized throughput* for medium-priority packets appears to be higher under the strict priority queuing algorithm for loads $0.7 \leq \lambda \leq 0.8$ as compared to the "mild prioritization" scheme (approximately by 2%), while the situation quickly reverses for higher loads. This can be explained by considering that under strict priority queuing, normal-priority packets are favored over low-priority ones, with the latter constituting the 30% of the overall network traffic. Thus, when the network operates under strict priority queuing, it appears to have amble resources to service high-, medium- and normal-priority packets, obviously at the expense of low-priority ones (cf. Figure 9). Beyond this load range however, the servicing of high- and medium-priority packets (which are now greater in numbers) consumes most network resources, resulting in degraded service being offered to medium-priority packets. Similarly we can explain the fact that the throughput offered to medium-priority packets under the "mild prioritization" scheme is better than the one offered under the "no priorities" scheme: medium-priority packets (25% of the overall traffic) are offered twice the bandwidth allocated to low-priority packets (30% of the overall traffic) and the gains from this setting seem to surpass the losses incurred from giving higher bandwidth shares to high- and medium-priority packets.

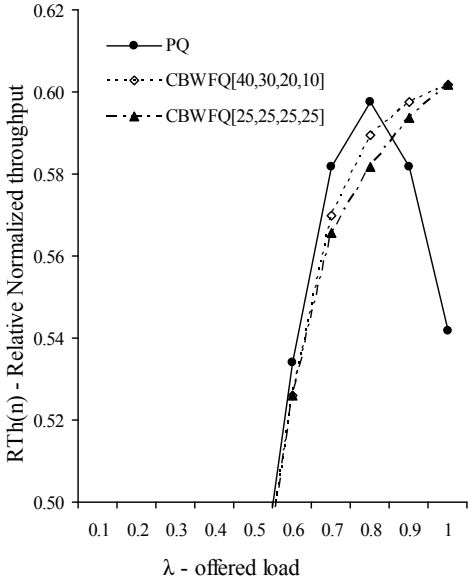

Figure 8. Relative normalized throughput for normal-priority packets vs. offered load

From the figures above we can conclude that the "mild prioritization" scenario offers considerable gains for low- and normal-priority packets, while the *normalized throughput* drops for high- and medium-priority packets can be considered tolerable.

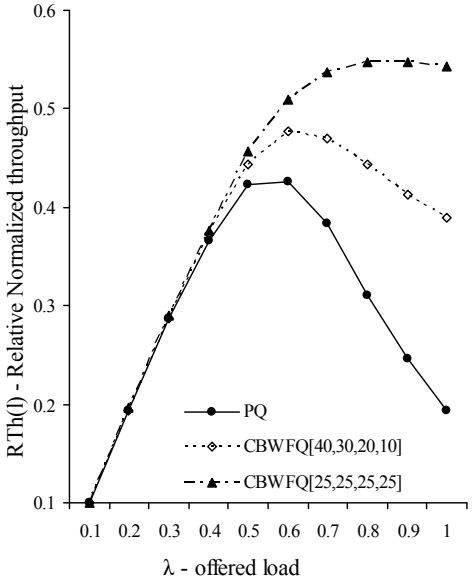

Figure 9. Relative normalized throughput for low-priority packets vs. offered load

Figure 10 illustrates the *normalized delay* for high-priority packets. The strict priority queuing scheduling algorithm offers the best delay, using however the CBWFQ algorithm under the "mild prioritization" scenario increases the delay only by 3.8% (λ=0.7) to 6.6% (λ=1); under the "no priorities" scenario, the delay metric deteriorates further, ranging from 7.5% (λ=0.7) to 11.3% (λ=1).

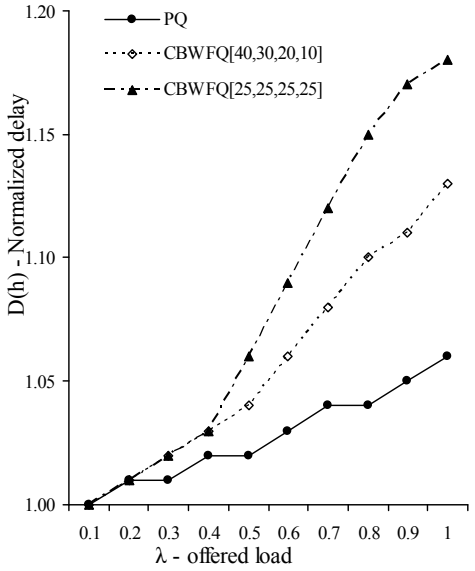

Figure 10. Normalized delay for high-priority packets vs. offered load

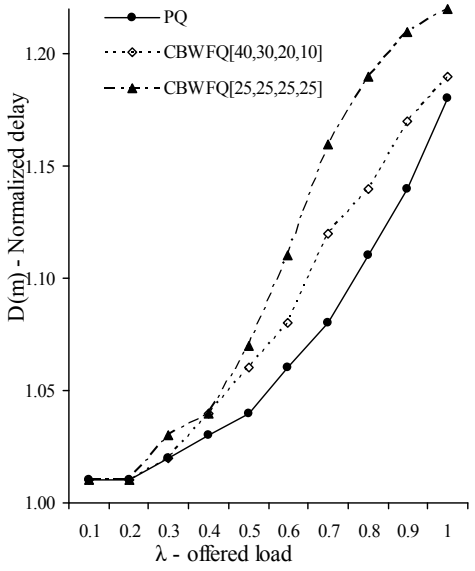

Figure 11. Normalized delay for medium-priority packets vs. offered load

For medium-priority packets (Figure 11), the deterioration of the *normalized delay metric* owing to the usage of the CBWFQ algorithm instead of strict priority queuing is even smaller: under the "mild prioritization" scenario the normalized delay increases by 0.8% (λ=0.7) to 3.7% (λ=1), while under the "no priorities" scenario, the corresponding increments are 7.4% (λ=0.7) to 3.3% (λ=1). Noticeably, the deterioration at full network load is smaller than the *normalized delay* increment for load λ=0.7: this is due to the fact that, under the strict priority queuing algorithm, the normalized delay increases steeply beyond load λ=0.7, since at this load range the network contains a considerable number of high-priority packets and the network resources are not adequate to optimally serve both high-and medium-priority packets.

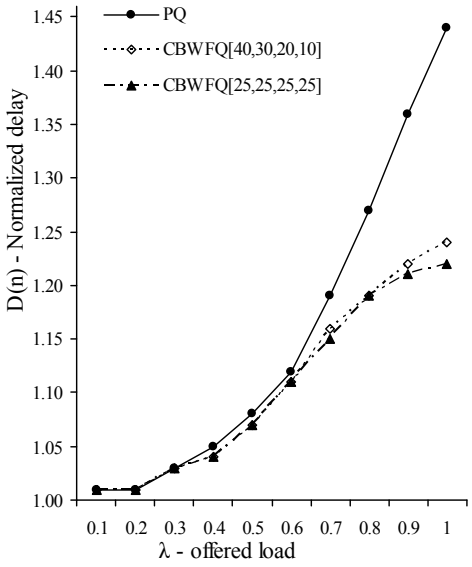

Figure 12. Normalized delay for normal-priority packets vs. offered load

Figure 12 depicts the *normalized delay* for normal-priority packets. We can notice than for load λ≥0.7 both CBWFQ scenarios offer significantly improved *normalized delay* for this priority class. In more detail, the "mild prioritization" scenario offers an improvement in the delay from 2.5% (λ=0.7) to 13.9% (λ=1), while in the "no priorities" scenario the improvement ranges from 3.3% (λ=0.7) to 15.2% (λ=1). Similar observations can be made for the *normalized delay* of low-priority packets (Figure 13), where under the "mild prioritization" scenario the normalized delay improves from 11.6% (λ=0.7) to 15.9% (λ=1), while under the "no priorities" scenario the improvements range from 18.4% (λ=0.7) to 23.3% (λ=1).

In overall, from the figures above we can conclude that the "mild prioritization" scenario offers considerable gains for low- and normal-priority packets, while the *normalized delay* drops for high- and medium-priority packets can be considered small (in all cases less than 7.5%).

Figures 14, 15, 16 and 17 illustrate the *universal performance factor* for packets belonging to the high, medium, normal and low priority classes, respectively. Recall that this metric combines the *normalized throughput* and *normalized delay* into a single metric, whose value

tends to zero when the network approaches its optimum operation.

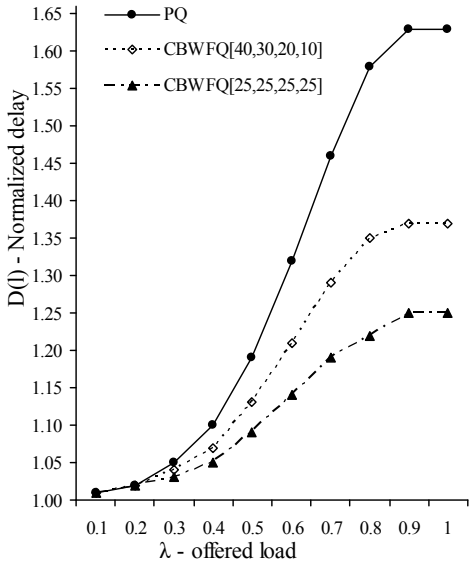

Figure 13. Normalized delay for low-priority packets vs. offered load

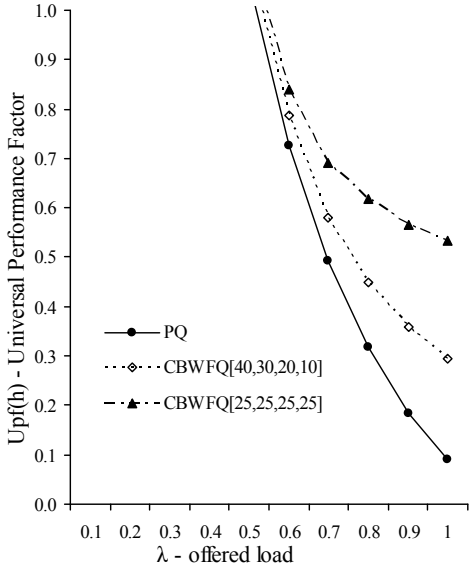

Figure 14. Universal performance factor for high-priority packets vs. offered load

For high- and medium-priority packets (Figure 14 and Figure 15), we can observe that the universal performance factor increases when shifting from the strict priority queueing algorithm to the "mild prioritization" scheme under CBWFQ, indicating that the QoS offered to these packet classes degrades. This is expected since both performance factors considered for the calculation of the *universal performance factor* (i.e. *relative normalized throughput* and *normalized delay*) have been found to deteriorate for these packet priority classes under the "mild prioritization" scheme, as compared to strict priority queuing. Under the "no priority" scheme, we can observe that the QoS offered to these packet classes degrades further.

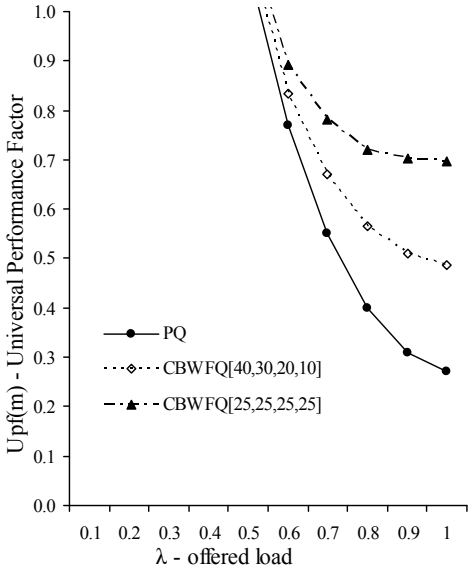

Figure 15. Universal performance factor for medium-priority packets vs. offered load

Regarding normal- and low-priority packets (Figure 16 and Figure 17), we can observe that switching from the strict priority queue algorithm to the "mild prioritization" scheme under CBWFQ leads to lower values of the *universal performance factor*, indicating that the QoS offered to packets of these priority classes increases. Again, this is expected since both the *relative normalized throughput* and *normalized delay* have been found to improve under the "mild prioritization" scheme. For load $\lambda = 0.7$ in particular, we can observe that for normal-priority packers the universal performance factor is *smaller* for strict priority queuing than CBWFQ/"mild prioritization", owing to the better throughput achieved by strict priority in this particular case, as discussed above. For load $\lambda = 0.8$, while strict priority queuing offers better *normalized throughput* to medium-priority packets than CBWFQ/"mild prioritization", strict priority queuing exhibits also considerably higher delays (cf. Figure 12); the final result is a (marginally) worse value for the strict priority queuing algorithm for the particular value of λ. Finally, the "no priorities" scheme yields even smaller values of the *universal performance factor* regarding low-priority packets, while for medium-priority packets the "no priorities" scheme results in higher values of the universal performance factor (i.e. worse QoS), owing to the fact that "mild prioritization" achieves better *normalized throughput* for normal-priority packets, as discussed above.

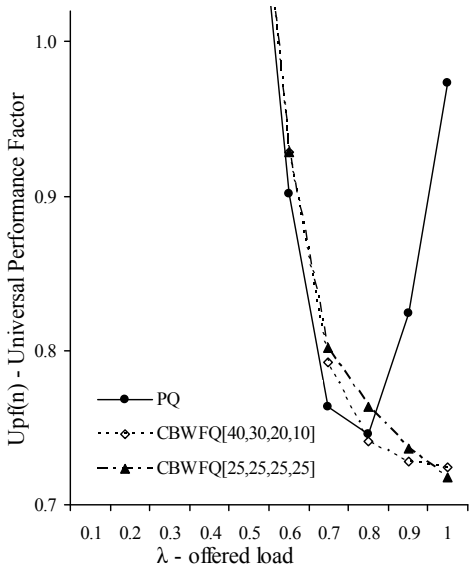

Figure 16. Universal performance factor for normal-priority packets vs. offered load

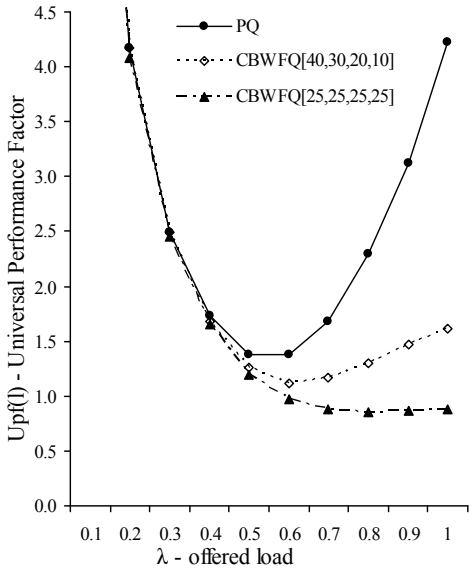

Figure 17. Universal performance factor for low-priority packets vs. offered load

V. CONCLUSIONS

In this paper we have addressed the performance evaluation of a quad-priority, single-buffered, 6-stage MIN, employing the class-based weighted fair queuing packet scheduling algorithm. We have evaluated through simulations the overall performance of the MIN and the quality of service offered to each priority class under different network loads and compared these results against the strict priority algorithm and the "no priority" scheme. The performance evaluation results show that the strict priority algorithm does offer the high- and medium-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 normal- and low-priority packets. Using a "mild prioritization" configuration, i.e. setting the high/medium/normal/low queue weights to 0.4/0.3/0.2/0.1 under a traffic mixture of 0.2/0.25/0.25/0.3 has been found to degrade the QoS offered to high- and medium-priority packets at a tolerable level, while it significantly improves the QoS offered to normal- and low-priority packets. Queue weights could further be tuned to obtain the desired QoS for all packet priority classes, always considering the ratio of packets in each priority class.

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 network sizes and load configurations, including hot-spot and burst loads, as well as different buffer sizes and handling schemes. In our future work we also intend to develop a closed form solution, providing thus an analytical model for single-buffered MINs incorporating the class-based weighted fair queuing algorithm on a quad-priority scheme. The provision of QoS with a reduced number of queues, as proposed in [36], [61] will be also considered.

VI. REFERENCES

[1] G.A. Abandah and E.S. Davidson, "Modeling the communication performance of the IBM SP2", in Proceedings of the 10th International Parallel Processing Symposium (IPPS'96), IEEE Q3 Computer Society Press, Hawaii, pp. 249-257, 1996.

[2] G. B. Adams and H. J. Siegel, "The extra stage cube: A fault-tolerant interconnection network for supersystems", IEEE Trans. on Computers, 31(4)5, pp. 443-454, May 1982.

[3] I. F. Akyildi, X. Wang, W. Wang. Wireless mesh networks: a survey. Computer Networks 47, pp. 445–487, 2005.

[4] 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 July 19, 2011.

[5] R.Y. Awdeh and H.T. Mouftah, "Survey of ATM switch architectures", Comput. Netw. ISDN Syst. 27 (1995), pp. 1567–1613, 1995.

[6] C. Bouras, J. Garofalakis, P. Spirakis, V. Triantafillou., "An analytical performance model for multistage interconnection networks with finite, infinite and zero length buffers", in Performance Evaluation 34(98) , pp. 169-182, 1998..

[7] Cisco Systems. QoS Scheduling and Queueing on the Catalyst 3550 Switches. http://www.cisco.com/en/US/tech/tk389/tk813/technologies_tech_note09186a00801558cb.shtml, accessed July 19, 2011.

[8] 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 accessed July 28, 2011.

[9] CISCO. Catalyst 6500 Release 15.0SY Software Configuration Guide. CISCO systems, 2011.

[10] CISCO. Catalyst 2960 Switch Software Configuration Guide. CISCO, 2010.

[11] CISCO. Catalyst 4500 Release IOS-XE 3.1.0 SG. Software Configuration Guide. CISCO systems, 2011.

[12] H. J. Chao. Next Generation Routers. Proceedings of the IEEE 90(9), September 2002.

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

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

[15] A. Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", Journal of Internetworking Research and Experience, V1, N1, pp 3-26, September 1990.

[16] DLink. D-Link™ DGS-1008D Gigabit Ethernet Switch manual. DLink 2007.

[17] A. Elwalid , D. Mitra, "Analysis, Approximations and Admission Control of a Multi-Service Multiplexing System with Priorities", Proceedings of IEEE INFOCOM '95, pp. 463-472, 1995.

[18] A. Elwalid , D. Mitra, "Design of generalized processor sharing schedulers which statistically multiplex heterogeneous QoS classes", Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99), pp. 1220 – 1230, 1999.

[19] S. Feit. Local area high speed networks.MTP, Indianapolis, 2000.

[20] G.F. Goke, G.J. Lipovski. "Banyan Networks for Partitioning Multiprocessor Systems" Procs. of 1st Annual Symposium on Computer Architecture, pp. 21-28, 1973.

[21] S.J. Golestani. A self-clocked fair queueing scheme for broadband applications. Proceedings of the 13th Conference on Networking for Global Communications (INFOCOM '94), pp. 636 – 646,1994.

[22] C. 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.

[23] A. 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.

[24] 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.

[25] Hewlet Packard. 3Com SuperStack 3 Switch 3800 – Overview. http://bizsupport1.austin.hp.com/bizsupport/TechSupport/Document.jsp?objectID=c02642521&printver=true, accessed July 19, 2011.

[26] Hewlet-Packard. HP ProCurve Secure Router 7000dl Series. Available at http://www.hp.com/rnd/pdfs/datasheets/ProCurve_Secure_Router_7000dl_Series.pdf accessed July 28, 2011.

[27] Hewlett-Packard, HP 2615-8-PoE Switch QuickSpecs. Hewlett-Packard, 2011.

[28] IEEE 802.1D Working Group. IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. IEEE 2004, http://standards.ieee.org/about/get/802/802.1.html

[29] IEEE 802.1Q Working Group. IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. IEEE 2005, http://standards.ieee.org/about/get/802/802.1.html

[30] Y.-C.Jenq, "Performance analysis of a packet switch based on single-buffered banyan network", IEEE Journal Selected Areas of Communications (83), pp. 1014-1021, 1983.

[31] J. Kim, T. Shin, and M. Yang, "Analytical modeling of a Multistage Interconnection Network with Buffered axa Switches under Hot-spot Environment", Procs. of PACRIM'07, 2007.

[32] S. Kumar, "Mathematical Modelling and Simulation of a Buffered Fault Tolerant Double Tree Network", International Conference on Advanced Computing and Communications ADCOM'07, Volume , Issue pp.422 – 433, 18-21 Dec. 2007.

[33] D.A. Lawrie. "Access and alignment of data in an array processor", IEEE Transactions on Computers, C-24(12):11451155, Dec. 1975.

[34] 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.

[35] T.Lin, L. Kleinrock, "Performance Analysis of Finite-Buffered Multistage Interconnection Networks with a General Traffic Pattern", Joint International Conference on Measurement and Modeling of Computer Systems, Proceedings of the 1991 ACM SIGMETRICS conference on Measurement and modeling of computer systems, San Diego, California, United States, Pages: 68 - 78, 1991

[36] A. Martinez, F.J. Alfaro, J.L. Sanchez, F.J. Quiles, J. Duato. A New Cost-effective Technique for QoS Support in Clusters. IEEE Transactions on Parallel and Distributed Systems. 18(2), pp. 1714 – 1726, 2007

[37] H. Mun and H.Y. Youn. "Performance analysis of finite buffered multistage interconnection networks", IEEE Transactions on Computers, pp. 153-161, 1994.

[38] 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, June 1999.

[39] Netgear. Quality of Service (QoS) on Netgear switches. Netgear, 2009.

[40] Network Working Group. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. IETF, 1998. http://tools.ietf.org/html/rfc2474

[41] Nortel Networks. Nortel Ethernet Switch 460/470 Overview — System Configuration. http://support.avaya.com/css/P8/documents/100099692, accessed July 19, 2011

[42] J.H. Patel. "Processor-memory interconnections for mutliprocessors", Procs. of 6th Annual Symposium on Computer Architecture. New York, pp. 168-177, 1979.

[43] B. Prabhakar, N. McKeown. On the speedup required for combined input- and output-queued switching. Automatica, Volume 35, Issue 12, pp.1909-1920, December 1999.

[44] G. Shabati, I. Cidon, and M. Sidi. Two priority buffered multistage interconnection networks. Journal of High Speed Networks, pp.131–155, 2006.

[45] J. Shortle, M. Fisher. Approximation for a two-class weighted fair queueing discipline. Performance Evaluation 67 (2010) 946-958.

[46] 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.

[47] 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.

[48] 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.

[49] W.R. Stevens, "TCP/IP Illustrated", Volume 1. The protocols, (10th Ed), Addison-Wesley Pub Company, 1997.

[50] 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.

[51] [J. Torrellas and Z. Zhang, "The performance of the cedar multistage switching network", IEEE Trans. Parallel Distrib. Syst. 8(4) , pp. 321–336, 1997.

[52] 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.

[53] D. 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.

[54] D. 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.

[55] D.C. Vasiliadis, G.E. Rizos, and C. Vassilakis. “Performance Analysis of blocking Banyan Swithces”, Procs. of CISSE 06, December, 2006.

[56] 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.

[57] 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.

[58] D.C. Vasiliadis, G.E. Rizos, C. Vassilakis, E. Glavas. “Routing and Performance Analysis of Double-Buffered Omega Networks Supporting Multi-Class Priority Traffic”, Proceedings of International Conference on Systems and Networks Communications ICSNC08, IEEE Press, 2008.

[59] 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.

[60] 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).

[61] A. M. Vicente, G. Apostolopoulos, F.J. Alfaro, J.L. Sanchez, F.J. Duato. Efficient Deadline-Based QoS Algorithms for High-Performance Networks. IEEE Transactions on Computers. 57(7), pp. 928-939, July 2008.

[62] C. A. Waldspurger, W. E. Weihl, “Lottery Scheduling: Flexible Proportional-Share Resource Management”, In Proceedings of symposim on Operating System Design and Implementation, November 1994.

[63] 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.

[64] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet-Switched Networks. ACM Transactions on Computer Systems, Vol 9, No. 2.,,pp. 101-124, May 1991.