Hierarchical Scheduling for DiffServ Classes

Mei Yang[†], Jianping Wang[‡], Enyue Lu[⋆], S. Q. Zheng[*]

[†] Department of Electrical and Computer Engineering, University of Nevada Las Vegas, Las Vegas, NV 89154

[‡] Department of Computer Sciences, Georgia Southern University, Statesboro, GA 30460

[⋆] Dept. of Mathematics and Computer Science, Salisbury University, Salisbury, MD 21801

[*] Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080

E-mail:[†]meiyang@egr.unlv.edu, [‡]jpwang@georgiasouthern.edu, [⋆]ealu@salisbury.edu, [*]sizheng@utdallas.edu

***Abstract*—Due to its simplicity and scalability, the differentiated services (DiffServ) model is expected to be widely deployed across the Internet. For each DiffServ compliant router, the scheduling algorithm is critical in implementing per hop behaviors (PHBs), according to which packets are forwarded. In this paper, we propose the hierarchical DiffServ scheduling (HDS) algorithm to support DiffServ classes on input-queued switches. The proposed HDS algorithm features in a hierarchical scheduling scheme that consists of two levels of schedulers. One level is the central scheduler which is designed to maximize the switch throughput by computing a maximal size matching between input ports and output ports. The other level is formed by input port schedulers which provide differentiated services by serving cells belonging to different classes dynamically. Using such a hierarchical scheme, the implementation complexity and the amount of information needs to be transmitted between input ports and the central scheduler are dramatically reduced compared with existing maximal weight matching based DiffServ scheduling algorithms. The tradeoff of its slightly worse delay performance is acceptable.**

I. Introduction

Differentiated Services (DiffServ) is proposed to meet different quality of service (QoS) requirements for various types of clients and network applications. The DiffServ model [1] is orientated toward edge-to-edge service across a single domain. It pushes the flow-based traffic classification and conditioning to edge routers of the domain. Core routers of the domain do not need to maintain per-flow state information, but only need to forward packets according to the per hop behavior (PHB) associated with each traffic class, which is identified by the DiffServ code point (DSCP) field in the header of each packet. The DiffServ model matches the heterogeneous feature of the Internet and it is capable of providing end-to-end QoS guarantees by bilateral agreements between neighboring domain owners [2]. Due to its simplicity and scalability, DiffServ is expected to be widely deployed across the Internet.

Currently, the IETF defines a set of PHBs which include Expedited Forwarding (EF) PHB, Assured Forwarding (AF) PHB group, and Best Effort (BE) PHB. The EF PHB provides low loss, low delay, low jitter, assured bandwidth, and end-to-end service through the DiffServ domain. The EF PHB is ideally suitable for voice over IP (VoIP), audio-, video- streaming, and other real-time applications. The AF PHB group provides services with minimum rate guarantee and low loss rate [3]. Four AF classes (AF1, AF2, AF3, and AF4) are defined and each class has three levels of drop precedence [3]. The level of forwarding assurance of an IP packet belonging to an AF class depends on the amount of resources allocated to the AF class, the current load of the AF class, and the drop precedence of the packet. AF PHBs are suitable for network management protocols, such as Telnet, SMTP, FTP, HTTP. All IP packets belonging to the BE class are not policed and are forwarded with the best effort.

The implementation of PHBs relies much on the scheduling and queuing schemes used in DiffServ complaint switches and routers. In order to provide premium service to EF traffic, packets belonging to EF class should be served prior to packets belonging to other classes. Meanwhile, to prevent the influence of damaging EF traffic to other traffic, the service rate (bandwidth) for EF traffic should be limited to its peak information rate (PIR). For each AF class, a minimum service rate, referred as committed information rate (CIR), should be guaranteed. On the other hand, to avoid starvation of BE traffic, backlogged BE queues should be served if excess bandwidth is available. In practice, we desire those scheduling and queuing schemes which are efficient in providing differentiated services for different traffic classes, with high throughput, and simple in implementation.

DiffServ supporting scheduling schemes proposed for output-queued (OQ) switches include priority queuing (PQ), weighted round-robin (WRR), PQWRR [4], and class-based queuing (CBQ) [5]. Among these algorithms, PQWRR is shown to have appealing performance and is simple in implementation. However, the aforementioned schemes are all based on unscalable OQ switch architectures. Compared with OQ switches, input-queued (IQ) switches are more scalable since they only need the switching fabric and memories to run at the line rate. In the literature, many QoS supporting scheduling algorithms have been proposed for IQ switches [6], [7], [8], [9], [10], [11]. Most of them are maximal weight matching (MWM)-based algorithms [12] with different definitions of the weight. Due to the lack of service reservation schemes, these algorithms cannot provide bandwidth or delay guarantee for each traffic class. The distributed mutlilayered scheduler (DMS) proposed in [13] for multistage switches can provide delay bounds for EF flows, and guaranteed bandwidth for AF flows. However, the complex structure of DMS and maintenance of per-flow queues prevent its practical use. In [14], we proposed the dynamic DiffServ scheduling (DDS) algorithm, which provides minimum bandwidth guarantees for EF and AF traffic and fair bandwidth allocation for BE traffic. DDS is also a MWM-based algorithm, for which the implementation com-

plexity is still high.

In this paper, we focus our study on efficient and practical DiffServ supporting scheduling algorithms for IQ switches. Extending the idea of hierarchial scheduling [8], we propose the hierarchical DiffServ scheduling (HDS) algorithm which provides minimum bandwidth guarantees for EF and AF classes and fair bandwidth allocation for BE class as the DDS algorithm but with a much simpler implementation.

To reduce the implementation complexity of the scheduler, we separate the tasks of providing differentiated services and maximizing switch throughput. The proposed HDS algorithm features in a hierarchical scheduling scheme which consists of two levels of schedulers. One level is the central scheduler which is designed to maximize the switch throughput by computing a maximal size matching (MSM) between input ports and output ports. The other level is input port schedulers which provide differentiated services by serving cells belonging to different classes dynamically. In light of the idea of exhaustive matching [15], the central scheduler employs a three-phase exhaustive MSM algorithm. At the granted input port, the service policy changes according to the bandwidth utilization at the destined output port such that minimum bandwidth guarantees for EF and AF classes and fair bandwidth allocation for BE class are provided. Using such a hierarchical scheme, the implementation complexity of the HDS algorithm is dramatically reduced compared with existing MWM-based DiffServ scheduling algorithms. Through simulations, we also evaluate the delay/jitter performance of HDS and compare them with PQWRR and DDS.

The rest of the paper is organized as follows. Section II introduces the IQ switch architecture. Section III presents the HDS algorithm. Section IV discusses simulation results. Section V concludes the paper.

II. IQ Switch Architecture

Figure 1 shows an $N \times N$ IQ switch architecture which consists of N input/output ports and a central scheduler. We assume that the switch architecture is cell-based, which means that all IP packets arriving at the switch are segmented into fixed-size cells, transmitted through the switching fabric, and reassembled into original IP packets before they leave the switch. We also assume that time is slotted such that one cell slot is equal to the transmission time of one cell on the input/output line. To remove head-of-line (HOL) blocking, each input port maintains N groups of virtual output queues (VOQs), and each group of VOQs is used to buffer cells destined for an output port.

As shown in Figure 2, at input port I_i, VOQ group $Q_{i,j}$ is composed of K separate FIFO queues $Q_{i,j,k}$'s, where $Q_{i,j,k}$ is used to buffer cells belonging to traffic class k, $1 \leq k \leq K$, and destined for output port O_j, $1 \leq j \leq N$. For DiffServ model, we have $K = 6$ with $k = 1..6$ representing class of EF, AF1, AF2, AF3, AF4, and BE respectively. When a cell arrives at an input (port), it is classified based on its DSCP field and destination address, and buffered in the VOQ corresponding to its traffic class and output (port).

In each cell slot, a scheduling algorithm is needed to determine which N cells in the N^2K VOQs to be transmitted through the switching fabric.

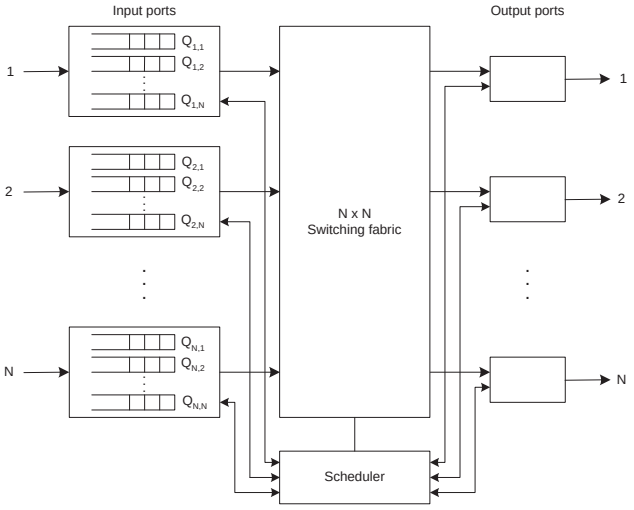

Fig. 1. The IQ switch architecture.

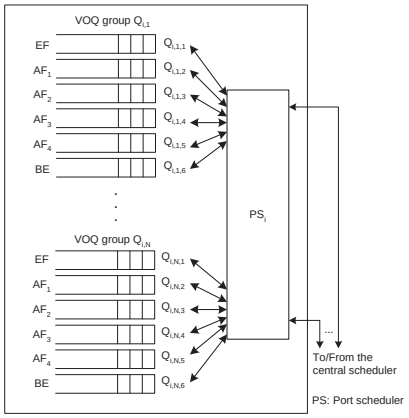

Fig. 2. Queuing and scheduling schemes at input port I_i.

III. The HDS Algorithm

Three factors need to be considered when designing a DiffServ supporting scheduling algorithm for IQ switches. First, to provide minimum bandwidth guarantees for EF and AF classes, the scheduling algorithm needs to consider the PIR for EF class and CIRs for four AF classes. Meanwhile, to avoid starvation of BE class, backlogged queues should be served if the excess bandwidth is available. Hence, class differentiation, bandwidth reservation and measurement schemes need to be introduced in the scheduling algorithm. Second, the switch throughput should be kept as much as possible. Third, the scheduling algorithm should be simple in implementation.

In order to reduce the implementation complexity of the scheduler, we propose the hierarchical DiffServ scheduling algorithm based on the idea of separating the tasks of providing differentiated services and maximizing switch throughput. The idea of hierarchical scheduling was first introduced in [8] to provide QoS guarantees for real-time traffic as well as high switch throughput. We extend the idea here to support DiffServ classes. The HDS algorithm features in a hierarchical scheduling scheme which consists of two levels of schedulers. One

level is the central scheduler which is designed to maximize the switch throughput by computing a maximal size matching between input ports and output ports. The other level is formed by input port schedulers which provide differentiated services by serving cells belonging to different classes dynamically.

A. Preliminaries

Before we present the HDS algorithm, we first introduce the bandwidth measurement schemes at each output port, which are similar to the DDS algorithm [14]. We use L to denote the bandwidth at each output link, which is divided into two categories, reserved bandwidth and excess bandwidth. To provide bandwidth guarantees for AF classes in a finer granularity and enforce smooth AF traffic, we introduce the time unit of $frame$, which is composed of T time slots. Each output port O_j, $1 \le j \le N$, maintains the following variables.

- $R_{j,k}$ denotes the reserved (guaranteed) bandwidth for class k, where $1 \le k \le K-1$. $R_{j,1}$ = PIR for EF class, $R_{j,k}$ = CIR for AF$(k-1)$ class, $2 \le k \le K-1$, and $\sum_{k=1}^{K-1} R_{j,k} \le 1$.
- $C_{j,k}$ denotes the cell counter for class k. $C_{j,1}$ counts the number of EF cells up to the current slot, and $C_{j,k}$, $2 \le k \le K-1$, counts the number of AF$(k-1)$ cells transmitted in the current frame. We set $C_{j,1} = 0$ at cell slot $t = 0$, and $C_{j,k} = 0$ at cell slot $t \bmod T = 0$ for $2 \le k \le K-1$.
- $S_{j,k}$ denotes the bandwidth utilization status for class k. $S_{j,k} = 1$ if $C_{j,1}/t < R_{j,1}$ for EF class or $C_{j,k}/T < R_{j,k}$ for AF$(k-1)$ class, $2 \le k \le K-1$; $S_{j,k} = 0$ otherwise.

At the beginning of each cell slot, each output port O_j, $1 \le j \le N$, sends S_j to the central scheduler. Each input port I_i, $1 \le i \le N$, collects the waiting time of the HOL cell of each non-empty VOQ $Q_{i,j,k}$ as $w_{i,j,k} = t - t'_{i,j,k}$, where $t'_{i,j,k}$ is the entering time slot of the HOL cell. We use a mapping function to map the weight value into the range of 0 to $2^{b_k} - 1$, where b_k is the number of bits used to represent the weight range of traffic class k. In this paper, we use a saturation function which is defined as follows.

$$f(w_{i,j,k}) = \begin{cases} w_{i,j,k} & \text{if } 0 \le w_{i,j,k} < 2^{b_k}, \\ 2^{b_k} - 1 & \text{otherwise.} \end{cases} \quad (1)$$

Each input port I_i only needs to send a $2N$-bit vector P_i to the central scheduler, where $P_{i,j} = 2$ if I_i has more than one EF cells in VOQ group $Q_{i,j}$, $P_{i,j} = 1$ if I_i has at least one cell in VOQ group $Q_{i,j}$, and $P_{i,j} = 0$ otherwise.

B. The HDS Algorithm

The HDS algorithm works in two stages.

Stage I: The central scheduler finds a maximal size matching in a three-phase exhaustive scheme iteratively. We assume that each input port I_i has an accept pointer a_i indicating the accept starting position, and each output port O_j has a grant pointer g_j indicating the grant starting position. Each iteration of stage I consists of the following three steps.

Step 1: **Request.** Each I_i sends a request to every O_j for which it has a queued cell.

Step 2: **Grant.** If an unmatched O_j receives any request, it selects one request to grant starting from the input port that g_j points to in a round-robin manner. For *the first iteration*, if $P_{i,j} = 2$ for some I_i, g_j is updated to i, otherwise, g_j is updated to one beyond the granted input port.

Step 3: **Accept.** If an unmatched I_i receives any grant, it selects one grant to accept starting from the output port that a_i points to in a round-robin manner. a_i is updated to the accepted output port.

After Stage I finishes, the central scheduler will send to each input port I_i an N-bit grant vector G_i, and S_j if there exists $G_{i,j} = 1$ for some j.

Stage II: For each input I_i that receives a non-zero grant vector (assuming that $G_{i,j} = 1$ for some O_j), if $\sum_{k=1}^{K-1} S_{j,k} f(w_{i,j,k}) \neq 0$, then it will select $Q_{i,j,k}$ such that $S_{j,k} = 1$ starting from $k = 1$ to $K-1$; otherwise, it will select $Q_{i,j,k}$ with $\max\{f(w_{i,j,k}) \mid f(w_{i,j,k}) > 0, 2 \le k \le K\}$.

Figure 3 illustrates an example of the exhaustive scheduling algorithm used at stage I for a 4×4 switch. At the beginning of the cell slot, grant pointers are set as $g_1 = 1$, $g_2 = 3$, $g_3 = 3$, and $g_4 = 2$, and accept pointers are set as $a_1 = 2$, $a_2 = 4$, $a_3 = 3$, and $a_4 = 1$. Given the request matrix P, in the request step, each input port I_i sends a request to each output O_j with $P_{i,j} > 0$ for $1 \le i, j \le 4$ as shown in Fig. 3 (a). As shown in Fig. 3 (b), in the grant step, each output grants one request starting from its grant pointer and updates its grant pointer accordingly. Notice that O_3 grants the request from I_3 and let g_3 stay at I_3 since $P_{3,3} = 2$. In the accept step, each input port accepts one grant starting from its accept pointer and updates its accept pointer to the accepted output port as shown in Fig. 3 (c). The generated grant matrix G is shown in the figure. Using such a pointer updating scheme, in the next cell slot, request from VOQ group $Q_{3,3}$ will continue to be favored, thereby serving EF traffic with the highest priority.

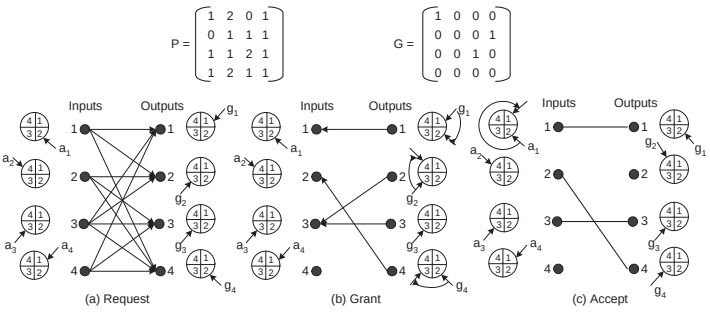

Fig. 3. An example of the exhaustive scheduling algorithm used at the central scheduler.

C. Hardware Implementation Scheme

The HDS algorithm distributes the selection of the highest weight request to each input port, hence simplifies the operation at the central scheduler. In each cell slot, the central scheduler only needs to find a maximal size matching. The number of iterations needed for the central scheduler to find a maximal size matching is at most N. Through simulations, we find that

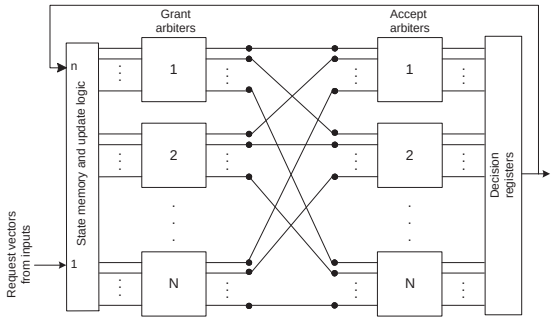

Fig. 4. Block diagram of the central scheduler.

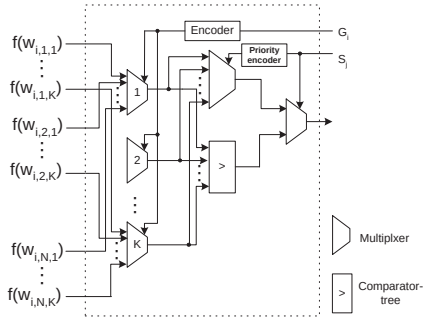

Fig. 5. Block diagram of the port scheduler at input port I_i.

on average $\log N$ iterations are adequate to achieve satisfying performance. To implement the central scheduler, one can use the scheduler architecture shown in Figure 4, in which each input/output is associated with an arbiter, which is responsible for selecting one out of N requests. Each arbiter can be implemented by the parallel round-robin arbiter proposed in [16], which has $O(\log N)$-gate delay. Hence, the first stage of the HDS algorithm can be implemented in $O(\log^2 N)$-gate delay.

As shown in Figure 5, each port scheduler majorly consists of K N-input multiplexers, one K-input multiplexer, and one K-input comparator-tree, which is responsible for selecting the maximum weight value among all traffic classes of the same VOQ group. Each port scheduler has $O(\log N + \log K \log b)$-gate delay, where $b = \max\{b_k \mid 1 \le k \le K\}$. The total delay of such an implementation of the HDS algorithm is $O(\log^2 N + \log K \log b)$-gate delay, which is faster than the implementation of the DDS algorithm, which has $O(\log^2 N \log b)$-gate delay [14]. The construction of the central scheduler and port schedulers is also simpler than that of the DDS scheduler.

In addition, the amount of information to be transmitted between each input port and the central scheduler in the HDS algorithm is much less than in the DDS algorithm. In each cell slot, in the HDS algorithm, each input port only needs to send $2N$ bits to the central scheduler and the central scheduler only needs to send $N + K$ bits back to each input port, while in the DDS algorithm, each input port needs to send NKb bits to the scheduler and the scheduler needs to send back NK bits to each input port.

IV. Performance Evaluation

In the following, we evaluate the performance of the HDS algorithm in two aspects: fairness and efficiency. Fairness is measured by received bandwidth percentage and efficiency is measured by average cell delay and delay jitter. Cell delay is the time that a cell spends in the switch counted in the number of cell slots. For EF traffic, we also consider its delay jitter performance, which is defined as the difference between the cell delays of two adjacent cells. To validate our evaluation, we compare the performance of the HDS algorithm with that of the DDS algorithm [14] and PQWRR algorithm [4].

We developed a cell-based simulator and conducted simulations assuming that all queue sizes are infinite. In our simulations, we consider bursty traffic arrivals using 2-state modulated Markov-chain sources [17]. Each source alternately generates a burst of full cells (all with the same destination) followed by an idle period of empty cells. The number of cells in each burst or idle period is geometrically distributed. Let $E(B)$ and $E(D)$ be the average burst length and the average idle length in terms of the number of cells respectively. Then, we have $E(D) = E(B)(1-\rho)/\rho$, where ρ is the load of each input source. We assume that the destination of each burst is uniformly distributed.

In all the simulations, we assume that the average cell arrival rates of EF class and AF classes to each output link are set as 18%, 24%, 20%, 16%, 12% by default. To ensure guaranteed service to EF traffic, we set its PIR a little more than its arrival rate [5], e.g. $R_{j,1} = 18\% \times 1.1 = 19.8\%$. The CIRs for AF1 through AF4 to each output port are set as 24%, 20%, 16%, 12% respectively. In the following simulations, we set the frame size as 1000 and $b_k = 4$ for all $1 \le k \le K$.

A. Bandwidth Allocation

We first evaluate the effectiveness of the HDS algorithm supporting fair bandwidth allocation when a link is overloaded. We assume a 4×4 switch, the average burst length $E(B) = 32$, and the number of iterations allowed for HDS is 4. We assume that output link 1 is the overloaded link and we vary the load to each VOQ group destined for output link 1 from 0.10 to 1.00.

Figure 6 and Figure 7 show the received bandwidth of each traffic class for PQWRR and HDS respectively. For a load below 0.25, the received bandwidth of each traffic class is able to keep up with its arrival rate for both schemes. However, for a load beyond 0.25, the received bandwidth of EF traffic by PQWRR still follows the arrival rate regardless of the limitation of its PIR. For a load beyond 0.30, due to the influence of damaging EF traffic, the received bandwidth of AF traffic by PQWRR is degrading dramatically, and BE traffic cannot get any service at all.

On the other hand, similar to DDS, HDS guarantees but limits the received bandwidth of EF traffic to its PIR, 19.8%, assures the CIR for each AF traffic, and avoids the starvation of BE traffic when the load is greater than 0.25. For example, when the load is at 0.40, EF traffic receives 19.8% bandwidth, AF1, AF2, AF3, AF4 traffic receives 25.70%, 21.37%, 16.60%, and 12.92% bandwidth respectively, and BE traffic receives 3.6% bandwidth.

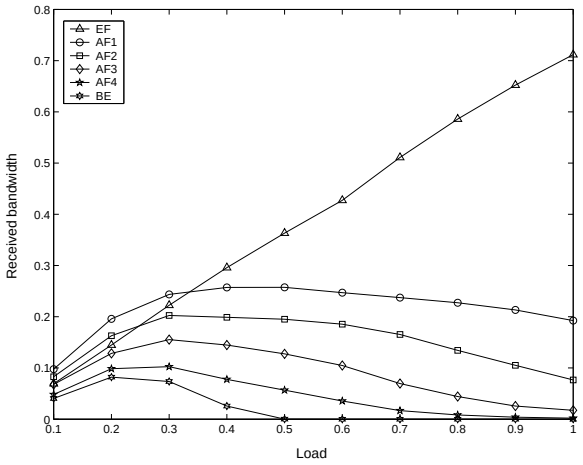

Fig. 6. Received bandwidth using PQWRR.

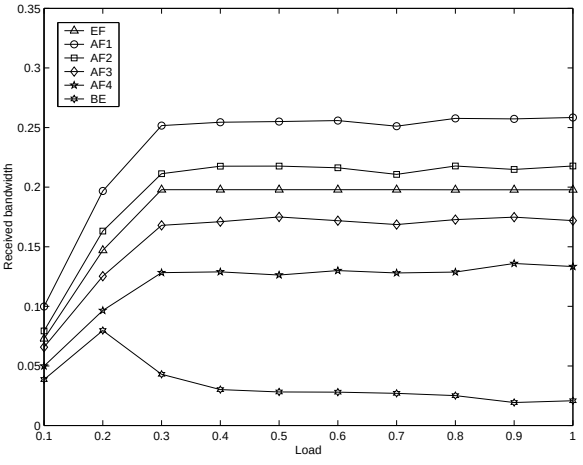

Fig. 7. Received bandwidth using HDS.

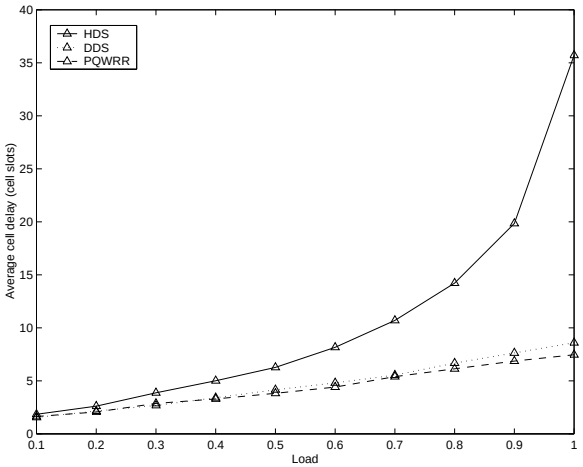

Fig. 8. Delay performance of EF traffic.

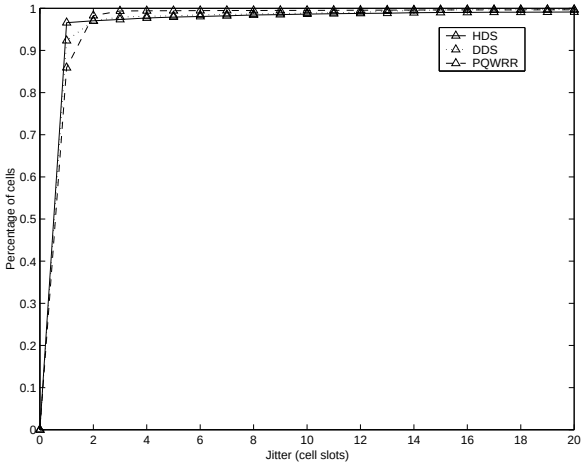

Fig. 9. EF jitter distribution.

B. Delay Performance

We then examine the delay performance of the HDS algorithm using simulations of a 16×16 switch under bursty arrivals assuming $E(B) = 32$ and the destination of each burst uniformly distributed. The number of iterations allowed for HDS is set as 4. Figure 8 shows the average cell delay vs. load of EF traffic for HDS, DDS, and PQWRR. The average cell delay of EF traffic using HDS is not as good as that using DDS and PQWRR. Figure 9 shows the jitter distribution of EF traffic at load 0.90 for HDS, DDS, and PQWRR. Using HDS, over 90% EF traffic has jitter less than 1 cell slot, which is comparable to DDS and PQWRR.

Figure 10 shows the average cell delay vs. load of AF1 and AF2 traffic for HDS, DDS, and PQWRR. The average cell delay of AF1 and AF2 traffic using HDS is slightly worse than that using DDS and PQWRR for most loads. For AF1 traffic, HDS tends to perform better than PQWRR for loads over 0.96. Figure 11 shows the average cell delay vs. load of AF3 and AF4 traffic for HDS, DDS, and PQWRR. For loads lower than 0.60, HDS performs close to PQWRR. With loads going up, the performance of HDS is degrading. Figure 12 shows the average cell delay vs. load of BE traffic for HDS, DDS, and PQWRR. For loads lower than 0.90, HDS performs better than DDS and PQWRR.

In the worst case, N iterations are needed for the central scheduler of HDS to find a maximal size matching. However, in reality, the number of iterations allowed in one cell slot is limited. Figure 13 shows the effect of the number of iterations allowed on the average cell delay of AF1 traffic using HDS. We can see that HDS with 2 iterations achieves significant performance improvement over HDS with 1 iteration. The performance of HDS with 4 iterations is very close to the performance of HDS with 16 iterations. Hence we set the number of iterations allowed as 4 for previous simulations on 16×16 switches.

V. Concluding Remarks

In this paper, we proposed the HDS algorithm to provide fair bandwidth allocation for DiffServ classes on IQ switches. The proposed HDS algorithm features in a hierarchical scheduling scheme which consists of two levels of schedulers: the central scheduler to maximize the switch throughput, and port schedulers to provide differentiated services by serving cells belonging to different classes dynamically. Using such a hierarchical scheme, the implementation complexity and the amount of information needs to be transmitted between each input port and the central scheduler are dramatically reduced compared

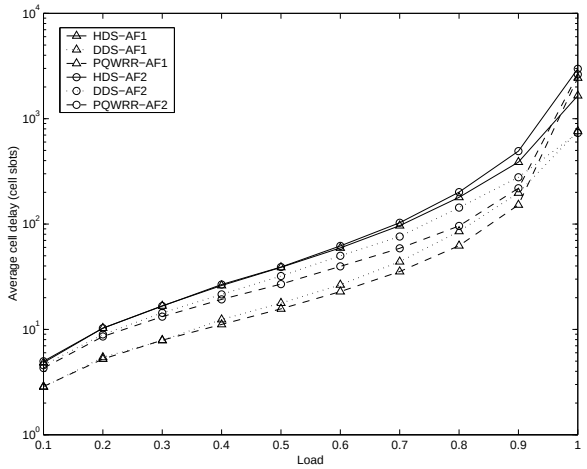

Fig. 10. Delay performance of AF1 and AF2 traffic.

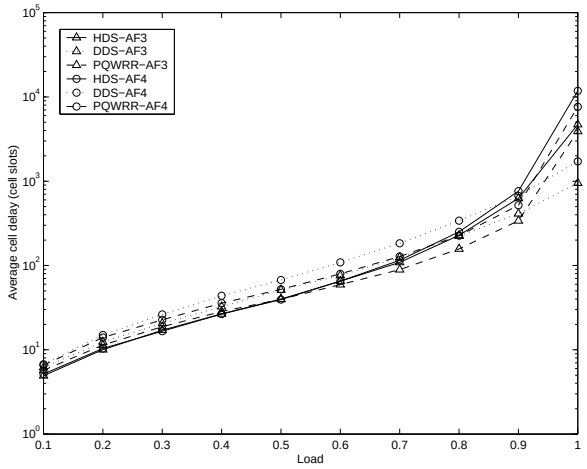

Fig. 11. Delay performance of AF3 and AF4 traffic.

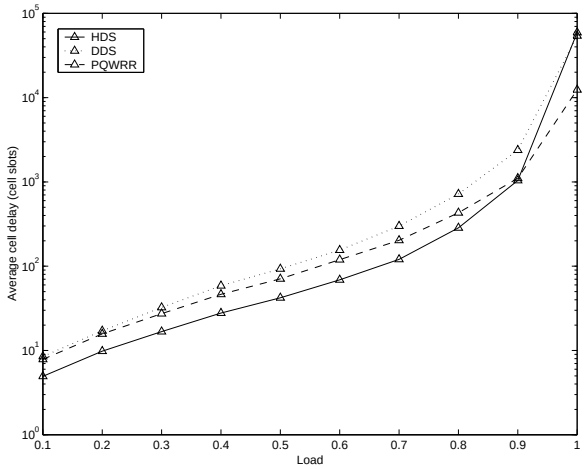

Fig. 12. Delay performance of BE traffic

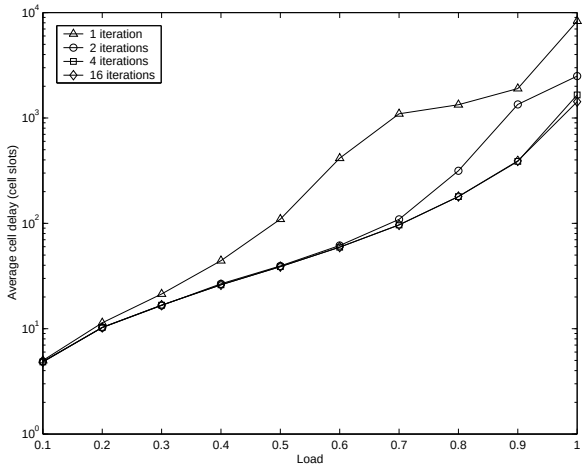

Fig. 13. Delay performance of AF1 traffic with different number of iterations allowed.

with existing MWM-based DiffServ scheduling algorithms. With bandwidth reservation and measurement scheme at output ports, HDS provides minimum bandwidth guarantees for EF and AF traffic with the reserved bandwidth and fair bandwidth allocation for BE traffic with the excess bandwidth. The tradeoff is the slightly worse delay performance for EF and AF traffic using HDS than that using DDS. Due to its simplicity, the HDS algorithm is very useful to implement the DiffServ model and it is applicable to other differentiated service models, such as the Olympic service [1].

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services", IETF RFC 2475, Dec. 1998.

[2] B. Carpenter, and K. Nichols, "Differentiated services in the Internet", in *Proc. IEEE*, vol. 90, no. 9, Sept. 2002, pp. 1479-1494.

[3] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group", IETF RFC 2597, Jun. 1999.

[4] J. Mao, W. M. Moh, and B. Wei, "PQWRR scheduling algorithm in supporting of DiffServ", in *Proc. ICC 2001*, vol. 3, pp. 679-684.

[5] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB group", IETF RFC 2598, Jun. 1999.

[6] C. Chen and M. Komatsu, "An adaptive scheduler to provide QoS guarantees in an input-buffered switch", in *Proc. ICC 2002*, vol. 2, pp. 1118-1122.

[7] A. Kam and K. Sui, "Linear complexity algorithms for QoS support in input-queued switches with no speedup", *IEEE J. Select. Areas Commu.*, vol. 17, no. 6, pp. 1040-1056, Jun. 1999.

[8] H. Kim, K. Kim, and Y. Lee, "Hierachical scheduling algorithm for QoS guarantee in MIQ switches", *IEEE Electronic Letters*, vol. 36, no. 18, pp. 1594-1595, Aug. 2000.

[9] S. Li and N. Ansari, "Provisioning QoS features for input-queued ATM switches", *Electronics Letters*, vol. 34, no. 19, pp. 1826-1827, Sept. 1998.

[10] R. Schoenen, G. Post and G. Sander, "Prioritized arbitration for input-queued switches with 100% throughput", in *Proc. IEEE ATM Workshop 1999*, pp. 253-258.

[11] M. Song and M. Alam, "Two scheduling algorithms for input-queued switches guaranteeing voice QoS", in *Proc. IEEE GLOBECOM 2001*, pp. 92-96.

[12] N. Mckeown, "Scheduling algorithms for input-buffered cell switches", Ph. D. Thesis, Univerity of California at Berkeley, 1995.

[13] F. Chiussi and A. Francini, "A distributed scheduling architecture for scalable packet switches", *IEEE J. Select. Areas Commu.*, vol. 18, no. 12, pp. 2665-2683, Dec. 2000.

[14] M. Yang, E. Lu, and S. Q. Zheng, "Scheduling with dynamic bandwidth share for DiffServ classes", in *Proc. ICCCN 2003*, pp. 319-324.

[15] Y. Li, S. Panwar, and H. J. Chao, "The dual round-robin matching with exhaustive service", in *Proc. IEEE HPSR 2002*, May 2002.

[16] S. Q. Zheng, M. Yang, J. Blanton, P. Golla, and D. Verchere, "A simple and fast parallel round-robin arbiter for high-speed switch control and scheduling", in *Proc. 45th IEEE MWSCAS*, 2002, pp. 671-674.

[17] N. Mckeown, "The *i*SLIP scheduling algorithm for input-queued switches", *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188-201, Apr. 1999.