Scheduling with Dynamic Bandwidth Allocation for DiffServ Classes

Mei Yang
Department of Computer Science
Columbus State University
Columbus, GA 31907
yang_mei@colstate.edu

Enyue Lu and S. Q. Zheng
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75083-0688
{enyue, sizheng}@utdallas.edu

***Abstract*— The diverse service requirements of emerging Internet applications foster the need for flexible and scalable IP quality-of-service (QoS) schemes. Due to its simplicity and scalability, DiffServ is expected to be widely deployed across the Internet. Though DiffServ supporting scheduling algorithms for output-queueing (OQ) switches have been widely studied, there are few DiffServ scheduling algorithms for input-queueing (IQ) switches. In this paper, we propose the dynamic DiffServ scheduling (DDS) algorithm for IQ switches to provide dynamic bandwidth allocation for DiffServ classes. The basic idea of DDS is to schedule EF and AF traffic according to their minimum service rates with the reserved bandwidth and schedule AF and BE traffic fairly with the excess bandwidth. We evaluate the performance of DDS under bursty traffic arrivals and compare it with PQWRR, an existing scheduling algorithm suitable for supporting DiffServ for OQ switches. Simulations results show that DDS provides minimum bandwidth guarantees for EF and AF traffic and fair bandwidth allocation for BE traffic. DDS also achieves the delay and jitter performance for EF traffic close to that of PQWRR and the delay performance for AF traffic better than that of PQWRR at high loads. Using comparator-tree based arbitration components, it is feasible to implement DDS in hardware at high speed.**

I. Introduction

The diverse service requirements of emerging Internet applications foster the need for flexible and scalable IP quality-of-service (QoS) schemes. Two IP QoS solutions proposed by the Internet Engineering Task Force (IETF) are integrated services (IntServ) and differentiated services (DiffServ). The IntServ model [2] provides QoS guarantees for individual flows through resource reservation and admission control mechanisms. The reservations are requested using the resource reservation protocol (RSVP). IntServ requires each IP router on the transmission path to reserve resources for each flow, maintain per-flow state information, and check each data packet in transit to ensure service requirements. The overhead of maintaining per-flow state information and checking each packet of the flow for resource management makes IntServ not scalable and not workable administratively [3].

To avoid the disadvantages of IntServ, the IETF proposed DiffServ, which is orientated toward edge-to-edge service across a single domain. DiffServ 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. 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 DiffServ model fits the heterogeneous feature of the Internet and it is capable of providing end-to-end QoS guarantees by bilateral agreements between neighboring domain owners [3]. Because of its simplicity and scalability, DiffServ is expected to be widely deployed across the Internet.

The implementation of PHBs relies much on the scheduling and queueing schemes used in switches and routers. There exist some DiffServ supporting scheduling schemes for output-queueing (OQ) switches, such as priority queueing (PQ), weighted round-robin (WRR), PQWRR [12], and class-based queueing (CBQ) [8], [11]. CBQ ensures explicit rate control for each traffic class by the rate control mechanisms functioned at two schedulers: the general scheduler and the link-sharing scheduler [6]. Compared with PQ and WRR, PQWRR delivers the minimum delay and jitter for EF traffic and provides better bandwidth allocation for AF traffic and BE traffic by priority scheduling of EF traffic and non-EF traffic, and weighted round-robin scheduling of AF traffic and BE traffic. With respect to implementation, PQWRR is more simple and practical than CBQ. Nevertheless, these schemes all assume OQ switch architectures which are not scalable for high line rates and/or large numbers of ports due to the speed limitation of the switching fabric and memories.

Compared with OQ switches, input queueing (IQ) switches are more scalable and practical since they only need the switching fabric and memories to run at the line rate. We hence focus our study on DiffServ supporting scheduling algorithms for IQ switches. Many QoS supporting scheduling algorithms have been proposed for IQ switches. Most of them are maximal weight matching (MWM) algorithms with different definitions of the weight, such as algorithms with the weight defined as a function of queue length (e.g. the SIMP algorithm [15], the LNQF algorithm [10], the worst-case LPF, and prioritized LPF algorithms [16]), algorithms with the weight defined as credits of bandwidth [9], and algorithms with the weight defined as time difference [4]. However, due to the lack of bandwidth reservation schemes, all these algorithms do not provide bandwidth or delay guarantee for each traffic class. Although the distributed mutlilayered scheduler (DMS) proposed in [5] for multistage switches can provide delay bounds for EF flows and guaranteed bandwidth for AF flows, the complex structure of DMS and maintenance of per-flow queues prevent its practical use.

In this paper, we propose the dynamic DiffServ scheduling (DDS) algorithm for IQ switches to provide dynamic bandwidth allocation for DiffServ classes. The DDS algorithm

assumes a cell-based IQ switch architecture, in which each input port maintains groups of virtual output queues (VOQs). A VOQ group is composed of multiple VOQs, each one of which is dedicated to holding cells that belong to a certain traffic class. The DDS algorithm finds a maximal weight matching in an iterative way with each iteration consisting of three steps: Request, Grant, and Accept (RGA). For the Grant step, the selection policy changes dynamically according to the bandwidth condition. If the reserved bandwidth is available, it serves EF or AF traffic first such that the peak information rate for EF class and the committed information rate for each AF class are guaranteed; otherwise, it serves non-EF traffic fairly such that BE traffic is not starved. Through simulations, we show that DDS provides minimum bandwidth guarantees for EF and AF traffic and fair bandwidth allocation for BE traffic. DDS also achieves the delay and jitter performance for EF traffic close to that of PQWRR and the delay performance for AF traffic better than that of PQWRR at high loads.

The rest of the paper is organized as follows. Section II briefly introduces the DiffServ architecture defined by IETF. Section III presents the switch architecture and the DDS algorithm. Simulation results and comparison with PQWRR are discussed in Section IV. Section V concludes the paper.

II. DiffServ Architecture

The DiffServ architecture puts complicated functionality at edge routers of the DiffServ domain and keeps core routers simple. Edge routers maintain all user traffic profiles; they classify, meter, and shape all incoming packets to ensure that the individual user traffic flow conforms to the service level agreement (SLA) specified by the network operator [1]. Flows are grouped into a small number of traffic classes. The DiffServ code point (DSCP) field of each packet is marked accordingly by edge routers. In contrast, core routers in the DiffServ domain do not need to maintain per-flow state information, but only need to perform differentiated aggregate treatment, known as per hop behavior (PHB), to each packet based on its DSCP field.

The DiffServ architecture standardized a set of PHBs: EF PHB, AF PHB group, and BE PHB. The EF PHB provides a low loss, low latency, low jitter, assured bandwidth, and end-to-end service through the DiffServ domain. This service is also called premium service, which appears to the end users like a point-to-point connection or a "virtual leased line". To guarantee the premium service, the implementation of EF PHB requires that the departure rate must be equal to or exceeding a configurable rate set by the network administrator. To prevent the influence of damaging EF traffic to other traffic, policing of the EF traffic requires strict enforcement of the peak information rate (PIR) such that traffic exceeding PIR must be discarded. 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 [7]. Four AF classes (AF1, AF2, AF3, and AF4) are defined and each class has three levels of drop precedence [7]. 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. A DiffServ compliant (DS) node must allocate a configurable, minimum amount of forwarding resources (buffers and bandwidth) to each AF class. We refer to the minimum bandwidth as committed information rate (CIR). An AF class may also be configured to receive more forwarding resources than the CIR when excess resources are available. Different excess bandwidth allocation schemes may be employed. 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 at the best effort.

In order to provide QoS guarantees for traffic aggregates, it is necessary to allocate enough network resources (buffers and bandwidth) at each DS node. In this paper, we assume that bandwidth for EF and AF traffic is pre-allocated. The minimum bandwidth allocated to EF traffic is equal to PIR, and the minimum bandwidth allocated to each AF traffic class is equal to its corresponding CIR. Bandwidth provisioning is out of the scope of this chapter.

III. Scheduling Algorithm

A. Switch Architecture

Figure 1 shows an $N \times N$ IQ switch architecture. We assume that all IP packets arriving at the switch are segmented into fixed-size cells, transmitted through the switching fabric, and recomposed back 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.

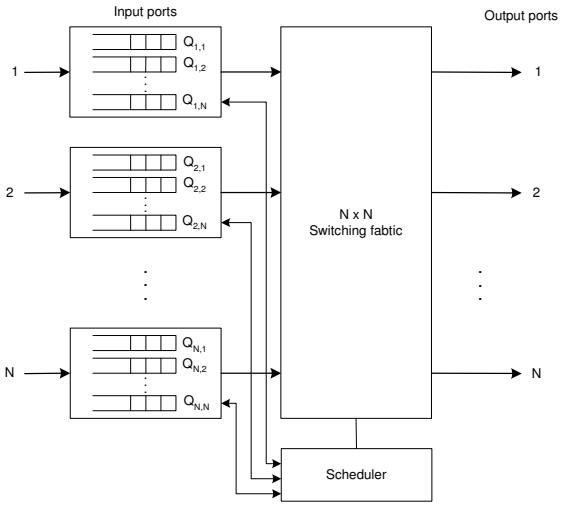

Fig. 1. The IQ switch architecture.

A VOQ group is composed of K VOQs, each dedicated to buffering cells of a DiffServ class. Figure 2 shows the queueing scheme used at input port I_i, $1 \leq i \leq N$, in which a separate FIFO queue $Q_{i,j,k}$ 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 the DiffServ model, we have $K = 6$ with $k = 1$ to 6 representing the classes 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 output port address, and buffered in the VOQ corresponding to its traffic class and output (port).

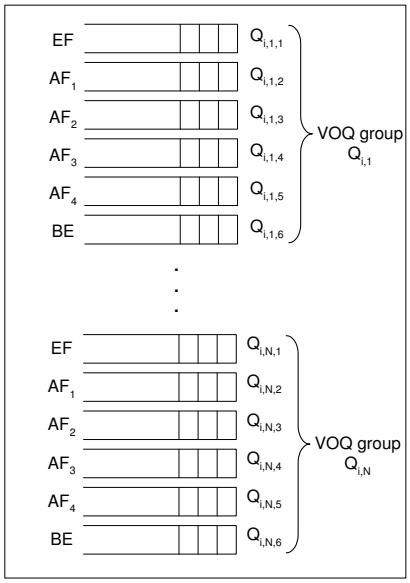

Fig. 2. Queueing scheme at input port I_i.

B. The DDS Algorithm

At the start of each cell slot, the switch scheduler collects the VOQ status from each input and decides a matching between inputs and outputs. In the following, we assume that scheduling in the current cell slot is based on the VOQ status of the previous cell slot, and switching in the current cell slot is based on the scheduling decision made by the previous cell slot. To provide minimum bandwidth guarantees for EF and AF classes, the scheduling algorithm needs to take into consideration the PIR for EF class and CIRs for four AF classes. To avoid starvation of BE class, backlogged queues should be served if the excess bandwidth is available. In the following, we will present our dynamic DiffServ scheduling (DDS) algorithm, that provides minimum bandwidth guarantees as well as fair bandwidth allocation for DiffServ classes.

We assume that the output link bandwidth in packets per second, denoted by L, is the same for all output links. The bandwidth of each output link is divided into 2 categories, the reserved bandwidth and the excess bandwidth (e.g., 90% as the reserved bandwidth and 10% as the excess bandwidth). The reserved bandwidth is further divided into 5 parts, each corresponding to the guaranteed bandwidth (service rate) for a non-BE DiffServ class. For example, the bandwidth reservation quotas for output link j are $R_{j,1}$ = PIR for EF, and $R_{j,k}$ = CIR for AF$(k-1)$, $2 \leq k \leq K-1$, and $\sum_{k=1}^{K-1} R_{j,k} \leq 1$. Note that, in the DDS algorithm, we represent the PIR (resp. CIRs) in the ratio of the original PIR (resp. CIRs) in packets per second to L. Accordingly, the service rate of each traffic class is represented in the ratio of the service rate in packets per second to L. The reserved bandwidth is allocated to EF and AF classes according to the class priority order as follows. For EF class, to guarantee the lowest delay and jitter performance, its bandwidth is always available as long as the number of transmitted EF cells is within the limit determined by its PIR. To guarantee the minimum service rate for each AF class, we assure that the bandwidth for any AF class is available if the number of transmitted cells belonging to that AF class does not exceed a limit determined by its CIR in a period of time, named a $frame$, which is composed of T time slots. The purpose of using frame is to provide bandwidth guarantees to AF classes to a finer grain as well as smooth AF traffic. We assume that each output port O_j maintains five counters, $C_{j,k}$'s, $1 \leq k \leq K-1$, for counting the number of cells transmitted in the current frame. We initialize $C_{j,1} = 0$ at the start of the first cell slot, and $C_{j,k} = 0$, for $2 \leq k \leq K-1$, at the beginning cell slot of each frame, i.e. at the start of any cell slot t such that $t \bmod T = 0$.

At the beginning of cell slot t, each input port I_i, $1 \leq i \leq N$, collects the waiting time of the head-of-line cells of all non-empty VOQs as $w_{i,j,k} = t - t'_{i,j,k}$, where $t'_{i,j,k}$ is the entering time slot of the head-of-line cell of $Q_{i,j,k}$. Since we assume that cells arriving in the current cell slot will be eligible for scheduling in the next cell slot, $w_{i,j,k} > 0$ for any non-empty $Q_{i,j,k}$, $w_{i,j,k} = 0$ for any empty $Q_{i,j,k}$, where $1 \leq i, j \leq N$ and $1 \leq k \leq K$. For each VOQ group $Q_{i,j}$, a weighted request vector $V_{i,j}$ is constructed as $(f(w_{i,j,1}), f(w_{i,j,2}), \cdots, f(w_{i,j,K}))$, where $f(w_{i,j,k})$ is defined by the following equation:

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

In (1), we assume that b_k is the number of bits used to represent the weight range of traffic class k. b_k is chosen as an appropriate value such that separation of weights of different classes is achieved and implementation complexity is minimized. f is a mapping function that maps the weight of traffic class k, $w_{i,j,k}$, into the range of 0 to $2^{b_k} - 1$. Other more complex mapping functions may be employed, such as those discussed in [15].

The DDS algorithm finds a maximal weight matching in an iterative way, with each iteration consisting of the following three steps.

Step 1: **Request.** Each unmatched input I_i sends request vectors $V_{i,j}$'s to their corresponding outputs.

Step 2: **Grant.** For each unmatched output O_j, once it receives at least one non-zero request vector, it grants one input as follows.

- If $C_{j,1}/t \leq R_{j,1}$ or $C_{j,k}/T \leq R_{j,k}$ for $2 \leq k \leq K-1$, it grants the input with $\max\{f(w_{i,j,k}) | f(w_{i,j,k}) > 0, 1 \leq i \leq N\}$ starting from $k = 1$ to $K-1$; otherwise, it grants the input with $\max\{f(w_{i,j,k}) \mid f(w_{i,j,k}) > 0, 1 \leq i \leq N, 2 \leq k \leq K\}$.
- If $f(w_{i',j,k'}) > 0$ is selected for some traffic class k' of input $I_{i'}$, it sends $I_{i'}$ a grant vector

with the k'-th entry equal to $f(w_{i',j,k'})$ and other entries equal to '0', and other inputs zero grant vectors (all entries of the vector are set as '0'). $C_{j,k'}$ is incremented for $1 \leq k' \leq K-1$ if the grant is accepted by $I_{i'}$ in the Accept step.

Step 3: **Accept.** For each input I_i that receives at least one non-zero grant vector, it selects the output with $\max\{f(w_{i,j,k}) \mid f(w_{i,j,k}) > 0, 1 \leq j \leq N\}$ starting from $k = 1$ to K. The accepted output is notified of the selection.

The measurement scheme at the output side equips the DDS algorithm with the ability of providing dynamic bandwidth allocation for DiffServ classes. As described in the Grant step, if the reserved bandwidth is available, the DDS algorithm allocates the reserved bandwidth to EF and AF traffic by serving the request with the highest weight value of the highest priority class; otherwise, it allocates the excess bandwidth to AF and BE traffic fairly by serving the request with the highest weight value among AF classes and BE class. Additionally, the DDS algorithm is starvation-free since the weight is generated based on the waiting time of the head-of-line cell and the excess bandwidth is shared by AF and BE traffic fairly.

Note that in Grant and Accept steps, there might be ties, i.e. requests with equal weights. Ties may exist among different traffic classes, or among different inputs/outputs. To ensure fairness, we break ties by making selections desynchornizedly. We set the selection starting position of each output or input in the static round-robin way. For example, at cell slot t, O_j starts its selection of inputs from $(j+t) \bmod N$ and its selection of classes from $(t \bmod (K-1)) + 1$, and I_i starts its selection of outputs from $(i+t) \bmod N$. Compared with breaking ties randomly [13], static round-robin is much easier to implement.

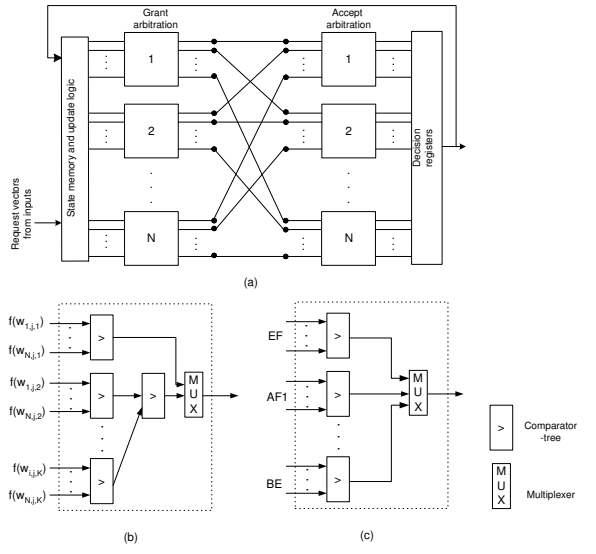

Fig. 3. (a) Block diagram of a DDS scheduler. (b) The grant arbitration component for output O_j. (c) The accept arbitration component for input I_i.

The core of the DDS algorithm is a maximal weight matching algorithm. The number of iterations needed to converge is at most N. However, through simulations, we find that on average $\log N$ iterations are adequate to achieve satisfying performance. To implement the DDS algorithm, one can use the scheduler architecture shown in Figure 3 (a), in which each input/output is associated with an arbitration component. As shown in Figure 3 (b) and (c), each arbitration component can be constructed by K copies N-input comparator-trees [13], each being used to find the maximum weight value for a class k, $1 \leq k \leq K$. One more comparator-tree is needed for each grant arbitration component to choose the maximum weight value of all classes. Each grant or accept arbitration component has $O(\log N \log b)$-gate delay, where $b = \max\{b_k \mid 1 \leq k \leq K\}$. Such an implementation of the DDS algorithm has $O(\log^2 N \log b)$-gate delay, which is feasible for high speed implementation.

IV. Performance Evaluation

We evaluate the performance of the DDS algorithm in two aspects: fairness and efficiency, where fairness is measured by the received bandwidth and efficiency is measured by the average cell delay and delay jitter. The cell delay is the queuing delay that a cell encounters in the switch. For EF traffic, we also consider its delay jitter performance, which is defined as the difference between the cell delays of two consecutive cells. To validate our evaluation, we compare the performance of the DDS algorithm with that of the PQWRR algorithm for OQ switches.

A cell-based simulator is developed and simulations have conducted assuming that all queue sizes are infinite. In our simulations, we consider bursty traffic arrivals using 2-state modulated Markov-chain sources [14]. 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 18%, 24%, 20%, 16%, and 12% respectively by default. To ensure guaranteed service to EF traffic, we set its PIR a little more than its arrival rate [8], e.g. $R_{j,1} = 18\% \times 1.1 = 19.8\%$. The CIRs for AF1 through AF4 to each output port are 24%, 20%, 16%, and 12% respectively. In the following simulations, we assume the frame size is 1000 and $b_k = 4$ for all $1 \leq k \leq K$.

First, we evaluate the effectiveness of the DDS algorithm supporting dynamic 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 DDS 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 4 and Figure 5 show the received bandwidth of each traffic class for PQWRR and DDS 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

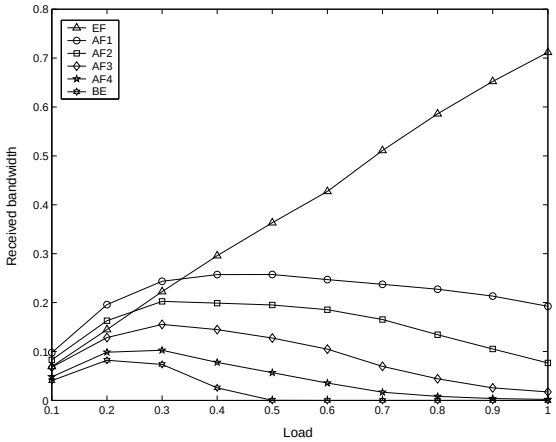

Fig. 4. Received bandwidth using PQWRR.

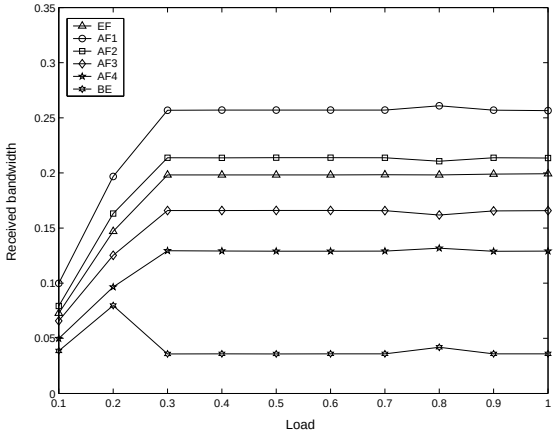

Fig. 5. Received bandwidth using DDS.

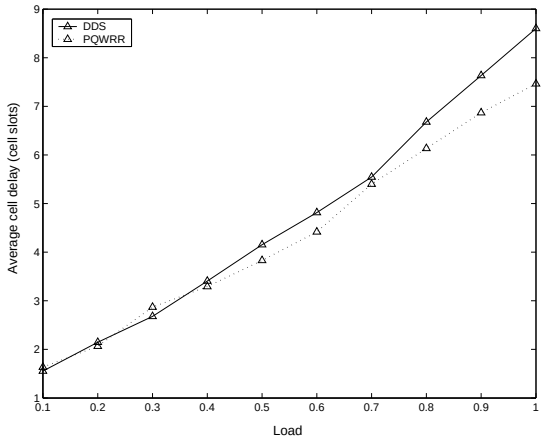

Fig. 6. Delay performance of EF traffic.

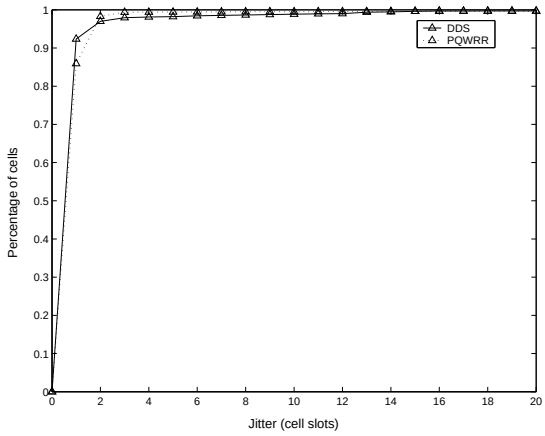

Fig. 7. EF jitter distribution.

rate without regarding to 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, DDS 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, the bandwidth received by EF, AF1, AF2, AF3, AF4, and BE traffic is 19.8%, 25.70%, 21.37%, 16.60%, 12.92%, and 3.6% respectively. Such a bandwidth distribution conforms to the design goal of DDS, which is to provide minimum bandwidth guarantees for non-BE classes and fair bandwidth allocation for BE class.

Next, we examine the delay performance of the DDS 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 DDS is set as 4. Figure 6 shows the average cell delay vs. load of EF traffic for DDS and PQWRR. The average cell delay of EF traffic using DDS is very close to that using PQWRR. Figure 7 shows the jitter distribution of EF traffic at load 0.90 for DDS and PQWRR. Using DDS, over 90% EF traffic has jitter less than 1 cell slot, which is comparable to PQWRR.

Figure 8 shows the average cell delay vs. load of AF1 and AF2 traffic for DDS and PQWRR. Figure 9 shows the average cell delay vs. load of AF3 and AF4 traffic for DDS and PQWRR. The average cell delay of each AF class using DDS is close to that using PQWRR for loads below 0.95. For loads over 0.95, DDS performs even better than PQWRR. The reason is that DDS uses a function of the waiting time as the weight but PQWRR uses the queue length as the weight. The tradeoff is that the average cell delay of BE traffic using DDS is relatively worse than that using PQWRR, as shown in Figure 10.

Though N iterations are needed for DDS to converge in the worst case, the number of iterations allowed in one cell slot is limited in reality. Figure 11 shows the effect of the number of iterations allowed on the average cell delay of AF1 traffic using DDS. We can see that DDS with 2 iterations achieves significant performance improvement over DDS with 1 iteration. The performance of DDS with 4 iterations is very close to the performance of DDS with 16 iterations. That is why we set the number of iterations allowed as 4 for previous simulations on 16×16 switches.

V. Conclusion

In this paper, we proposed the DDS algorithm to support dynamic bandwidth allocation for DiffServ classes on IQ

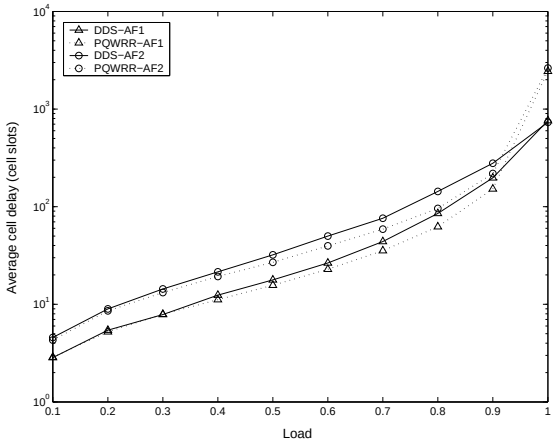

Fig. 8. Delay performance of AF1 and AF2 traffic.

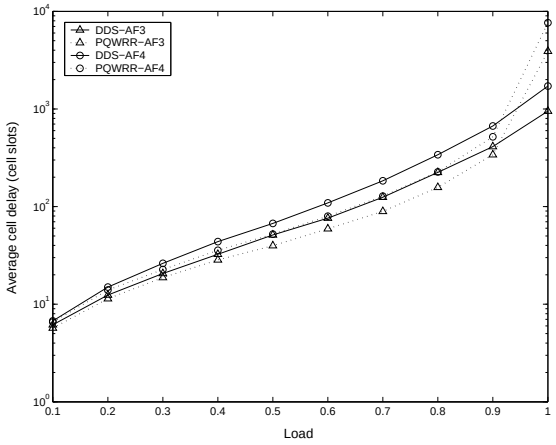

Fig. 9. Delay performance of AF3 and AF4 traffic.

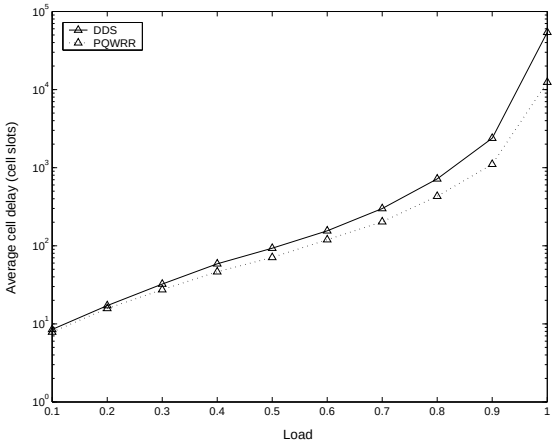

Fig. 10. Delay performance of BE traffic

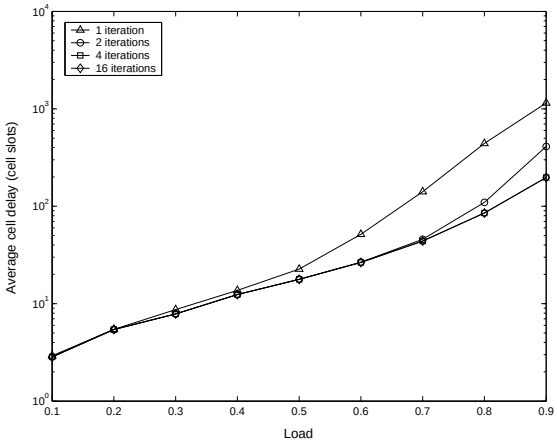

Fig. 11. Delay performance of AF1 traffic with different number of iterations allowed.

switches. The DDS algorithm provides minimum bandwidth guarantees for EF and AF traffic with the reserved bandwidth and fair bandwidth allocation for BE traffic with the excess bandwidth. In addition, DDS is starvation-free since it generates the weight based on the waiting time of the head-of-line cell instead of the queue length. Simulation results have shown that DDS achieves the delay and jitter performance for EF traffic close to that of PQWRR and the delay performance for AF traffic better than that of PQWRR at high loads. Since IQ switches are more scalable than OQ switches, DDS is more practical than PQWRR. The DDS algorithm is very useful to implement DiffServ model and it is applicable to other differentiated service models, such as the so-called Olympic service [7].

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] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview", IETF RFC 1633, 1994.

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

[4] 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.

[5] 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.

[6] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet switches", *IEEE/ACM Trans. Networking*, vol. 3, no. 4, pp. 365-386, Aug. 1995.

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

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

[9] 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.

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

[11] G. Mamais, M. Markaki, G. Politis, and I. S. Venieris, "Efficient buffer management and scheduling in a combined IntServ and DiffServ architecture: a performance study", in *Proc. ICATM 1999*, pp. 236-242.

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

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

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

[15] 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.

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