Bandwidth Guaranteed Multicast Scheduling for Virtual Output Queued Packet Switches

Deng Pan
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794, USA

Yuanyuan Yang
Dept. of Electrical and Computer Engineering
State University of New York
Stony Brook, NY 11794, USA

ABSTRACT

Multicast enables efficient data transmission from one source to multiple destinations, and has been playing an important role in Internet multimedia applications. Although several multicast scheduling schemes for packet switches have been proposed, they usually consider only short delay and high throughput but not bandwidth guarantees. However, fair bandwidth allocation is critical for the quality of service (QoS) of the network, and is necessary to support multicast applications requiring guaranteed performance services, such as online audio and video streaming. This paper addresses the issue of bandwidth guaranteed multicast scheduling on virtual output queued (VOQ) switches. We propose the Credit based Multicast Fair scheduling (CMF) algorithm, which aims at achieving not only short multicast latency but also fair bandwidth allocation. CMF uses a credit/balance based strategy to guarantee the reserved bandwidth of an input port on each output port of the switch. It keeps track of the difference between the reserved bandwidth and actually received bandwidth, and minimizes the difference to ensure fairness. Moreover, CMF supports multicast scheduling by allowing a multicast packet to send transmission requests to multiple output ports simultaneously. As a result, a multicast packet has more chances to be delivered to all its destinations in the same time slot, and thus shortens its multicast latency. Extensive simulations are conducted to compare the performance of CMF with other existing scheduling algorithms, and the results demonstrate that CMF achieves the two design goals: short multicast latency and fair bandwidth allocation.

Keywords: Multicast, fair scheduling, VOQ switch.

I. INTRODUCTION

Multicast enables data to be efficiently transferred from one source to multiple destinations, and has been playing an important role in Internet multimedia applications [1], such as teleconference, distance learning, and video on demand services. Although one multicast packet can be handled as multiple copies of a unicast packet, it is desired that multicast scheduling and switching are supported at the router/switch level to save network bandwidth and reduce multicast latency. In this paper, we consider multicast scheduling on packet switches. Such a switch can be used as a crossconnect in an intermediate router or an edge router of a wide area communication network. It can also provide high speed interconnections among a group of processors in a parallel and distributed computing system.

Packet switches can be divided into different categories based on where the blocked packets are queued. An output queued switch, as shown in Fig.1(a), buffers packets at their destination output ports, and is able to achieve 100% throughput. However, since there is no buffer at the input side, if multiple input ports have packets arriving at the same time that are destined to the same output port, all the packets must be transmitted simultaneously. Therefore, in order for an $N \times N$ output queued switch to work at full throughput, the switching speed of the internal fabric and the receiving speed of the output port must be N times faster than the sending speed of the input port. This deficiency makes output queued switches difficult to scale, especially when the switch has a large number of input ports or the speed of a single input port increases to gigabit/s [2] [3].

On the contrary, an input queued switch stores blocked packets at the input side, and therefore gets rid of the N speedup requirement. The single input queued switch, as in Fig.1(b), has a first-in-first-out (FIFO) queue at each input port to store the incoming packets. Because only the packets at the head of line (HOL) of each input queue can participate in the scheduling, the packets behind the HOL packet suffer from the "head of line" blocking, which means that even though their destination output ports may be free, they cannot be scheduled to transfer because the HOL packet is blocked. The HOL blocking severely affects the maximum throughput of the single input queued switch [4]. An efficient yet simple buffering strategy to remove the HOL blocking is to adopt the virtual output queued (VOQ) structure, as shown in Fig.1(c). A VOQ switch maintains N logically separate FIFO queues for buffering packets destined to the N different output ports. The HOL blocking is eliminated because a packet cannot be held up by another packet to a different output port. The traditional VOQ structure buffers packets to different destinations in different queues. However, since a multicast packet may be destined to multiple output ports, it has $2^N - 1$ possible destinations. This means that a VOQ switch for multicast traffic needs to maintain $2^N - 1$ separate queues at each of its input ports, which is obviously infeasible, especially for a large N.

The combined input output queued (CIOQ) switch, shown

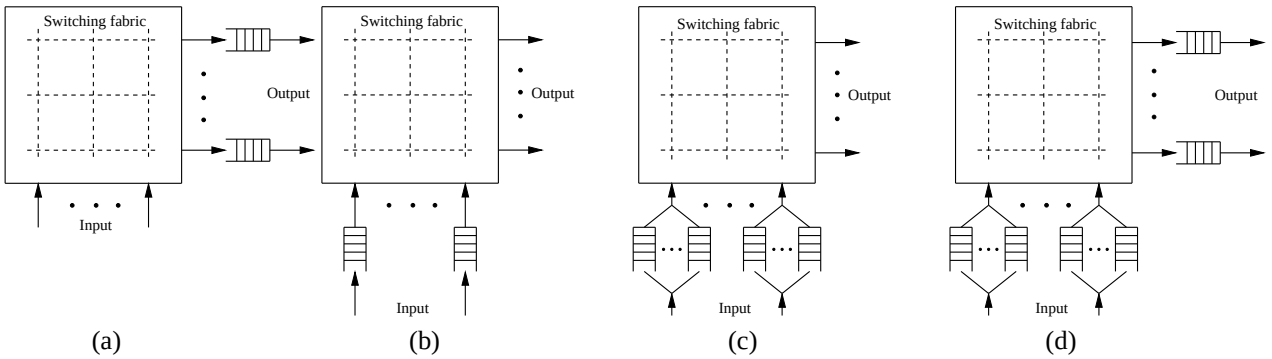


Fig. 1. Packet switches can be divided into different categories by the location where the blocked packets are buffered. (a) Output queued switch. (b) Single input queued switch. (c) Virtual output queued switch. (d) Combined input output queued switch.

in Fig.1(d), extends the VOQ switch by adding the speedup capability to the switching fabric. As a result, an output port may receive more than one packet in a single time slot, and needs buffer space to save the extra packets. [22] proves that, for a CIOQ switch with a speedup of 2, special algorithms can be designed to precisely emulate an output queued switch employing a wide variety of scheduling algorithms. However, because of the high complexity, these algorithms are only of theoretical interest and are not practical for high speed implementations at this time.

Due to its efficient hardware implementation, the input queued switch has been the main focus in the networking community, and several schemes are proposed to schedule multicast traffic on the input queued switch, see, for example, TATRA [5], ESLIP [23], and FIFOMS [7]. Existing multicast scheduling algorithms usually aim to achieve short delay and high throughput, without considering fair bandwidth allocation. In other words, the algorithms are not able to protect normal users from being affected by ill-behaved users. However, the quality of service (QoS) has become a main concern for the design of modern routers/switches, and it is a necessity in order for the network to provide not only best effort services but also guaranteed performance services. Bandwidth guaranteed fair scheduling on shared output links has been well studied, and a large number of algorithms have been proposed [9] - [15]. These algorithms can be easily applied to output queued switches to achieve fair bandwidth allocation, but as mentioned earlier, output queued switches are expensive to implement due to the speedup requirement. Therefore, some efforts [16] [17] [18] have been made to apply these algorithms to input queued switches mainly for scheduling unicast traffic, and positive results have been obtained.

The objective of this paper is to design a multicast fair scheduling algorithm for VOQ switches, to achieve not only short multicast latency but also fair bandwidth allocation. To be more specific, we consider an $N \times N$ VOQ switch with a crossbar as its switching fabric, which has built-in multicast capability and is able to simultaneously send a packet from one input port to multiple output ports in the same time slot. The algorithm should schedule the packets in such a way that the reserved bandwidth of an input port on each output port is guaranteed, and multicast packets are efficiently transmitted with short latency. We assume that the switch internally operates on fixed length packets in a synchronous time slot mode. As analyzed in [23], fixed length packet scheduling has significant advantages over variable length packet scheduling, and is adopted by most of the implemented high speed switches, such as Cisco 12000 GSR [23], Tiny Tera [24], and AN2 [3]. For variable length packets, they can be segmented into fixed size units upon arrival, transferred through the switch, and then reassembled into the original packets before departure.

In this paper, we propose an algorithm called *Credit based Multicast Fair scheduling (CMF)*. CMF uses a credit/balance based strategy to guarantee the reserved bandwidth of an input port on each output port. It keeps track of the difference between the bandwidth that an input port receives in the ideal fairness model and that in the algorithm, and minimizes this difference to ensure fairness. Moreover, CMF supports multicast scheduling by allowing a multicast packet to send transmission requests to multiple output ports simultaneously. Thus, the multicast packet has more chances to be delivered to all the destinations in the same time slot, and shortens its multicast latency. We also conduct simulations under both multicast traffic and unicast traffic to compare the performance of CMF with other existing scheduling algorithms, including multicast scheduling algorithms without bandwidth guarantees and unicast fair scheduling algorithms. The results demonstrate that CMF fulfills the design objectives: short multicast latency and fair bandwidth allocation.

The rest of the paper is organized as follows. Section II reviews some existing schemes for bandwidth guaranteed fair scheduling. Section III describes the multicast VOQ structure associated with the CMF algorithm. Section IV defines an ideal multicast fair scheduling model based on the output queued switch, which is used as the reference system. Section V presents the Credit based Multicast Fair scheduling algorithm. In Section VI, we use simulations to evaluate the performance of CMF. And finally section VII concludes the paper.

II. Related Work

In this section, we give a brief review of the work that has been done on the issue of bandwidth guaranteed scheduling for shared output links and input queued switches.

A. Bandwidth Guaranteed Scheduling on Shared Output Links

A lot of schemes have been proposed for bandwidth guaranteed fair scheduling on shared output links, as in the case that several flows share the same outgoing gateway. These algorithms can be classified into three types: (1) Time stamp based. Time stamp based fair schedulers, such as *WFQ* [9] and WF^2Q [10], compute time stamps for each packet upon its arrival, and schedule packets in the order of the computed time stamps. They usually provide excellent fairness guarantees and perfectly emulate the ideal fairness models, such as GPS [8]. However, due to the operation to sort packets in the order of their time stamps, time stamp based schedulers have high time complexity. (2) Round robin based. The scheduling principle of round robin schedulers, such as *DRR* [11] and *SRR* [12], is to serve the flows one by one, so that each flow has equal opportunity of consuming bandwidth. Round robin based fair schedulers achieve O(1) time complexity, but have poor delay bounds, as each flow has to wait for all other flows before transmitting the next packet. (3) Combination of both. Some recently proposed algorithms, such as *BSFQ* [13] and *Stratified Round Robin* [14], attempt to obtain the tight delay bound of time stamp based schedulers as well as the low time complexity of round robin based schedulers. They usually adopt a basic round robin like scheduling policy plus time stamp based scheduling on a reduced number of units. These schedulers improve the time complexity by reducing the number of items that need to be sorted, but they still have long worst case delay due to the round robin nature.

By running the algorithm at each output port, the above algorithms for shared output links can be easily applied to output queued switches to provide fair bandwidth allocation.

B. Bandwidth Guaranteed Scheduling on Input Queued Switches

There have also been some attempts to implement bandwidth guaranteed fair scheduling on input queued switches. *WPIM* [16] improves upon *PIM* [3] by introducing a bandwidth enforcement mechanism to provide probabilistic bandwidth guarantees for input-output connections. Based on the reservation, every input flow is assigned a quota that can be used in a frame with a constant number of slots, and the algorithm works by masking out from the matching process the flows that have consumed their quotas in the current frame. *iFS* [17] adapts *WFQ* [9], a time stamp based fair scheduler for shared output links, to VOQ switches. iFS uses a grant-accept two stage iterative matching method, and uses the virtual time as the grant criterion so as to emulate the GPS [8] ideal model at each output port. Similarly, *iDRR* [18] is the application of *DRR* [11], which is a round robin based fair scheduling algorithm for shared output links, to VOQ switches. iDRR uses the round robin principle in its iterative matching steps, and thus is able to make fast arbitration. Also, the feature that a matched pair can keep the status until the assigned quota is used up reduces the iterative rounds needed for convergence.

All these algorithms can be used to provide fair bandwidth allocation for scheduling on VOQ switches. However, none of them particularly takes multicast traffic into consideration, and as a result, their performance under mixed multicast/unicast traffic has the potential to be improved. *mFS* [17] extends iFS to schedule multicast traffic. It uses counters to record the number of transmitted packets to ensure fair bandwidth allocation. Unfortunately, mFS is built on the traditional VOQ switch structure. As discussed earlier, the traditional VOQ switch buffers packets on a per flow basis, and needs to maintain $2^N - 1$ separate queues at each input port in order to handle multicast traffic, which is not practical.

III. Multicast VOQ Switch

In this section, we describe the multicast VOQ switch structure that our proposed CMF algorithm is based on. Since the VOQ switch does not require speedup as the output queued switch, and also removes the HOL blocking that limits the maximum throughput of the single input queued switch, it is the preferred structure for packet switches. However, the traditional VOQ switch does not suit for multicast traffic. In the following, we describe a scheme for organizing packets in the input buffers of a multicast VOQ switch, so that the number of queues at each input port can be reduced from exponential ($2^N - 1$) to linear (N).

In general, the information that a packet carries can be viewed as including two parts. The first part is the destination address information, which is used by the switch to make scheduling decisions, i.e., deciding for each input port when and which output port its HOL packet should be sent to. The second part of the information is the payload data, which is the content to be forwarded to the destination output ports. When the switch handles only unicast traffic, where the payload data of a packet need to be sent only once from an input port to a single output port, it is natural to combine the two parts into a single unit and use it for both scheduling and transmission. However, when multicast traffic is involved, a packet may need to be sent to multiple output ports. Although the destinations are different, the data content to be sent is the same. Therefore, there is no need to store multiple copies of the same data content. A more efficient way would be to store the address and data content of a packet separately: the data are stored once and used for all destination addresses of the packet.

Two different types of cells are used to store the two parts of a packet: the data cell to store the payload content of the packet, and the address cell to store the destination information of the packet.

A data cell is created to store the data content when a new packet arrives at the switch. Its data structure can be described as follows:

```
DataCell {
```

```
    binary payloadData;
    int fanoutCounter;
}
```

The payloadData field stores the data content of a packet. Since we assume that the switch operates on fixed size packets, it can be implemented as a fixed size field. The fanoutCounter field records the number of destination output ports that the payloadData is going to be sent to. When a packet arrives at the switch, the fanoutCounter field of its data cell is equal to the fanout of the packet. As the payloadData is sent to part or all of the destinations of the packet, the number in the fanoutCounter field is decremented accordingly. When it becomes zero, it means that all the destination output ports have been served, and therefore the data cell can be destroyed to return the buffer space to the switch.

The address cell stores the destination address information of a packet. Specifically, an address cell represents one of the destination output port of the packet, and serves as a place holder in the virtual output queue corresponding to that output port. When a new packet with fanout k enters the switch, k address cells are created for these destination output ports. The data structure of an address cell can be described as follows:

```
AddressCell {
    int timeStamp;
    pointer pDataCell;
}
```

The timeStamp field records the arrival time of the packet that the address cell is related to. The field has extra precision digits to differentiate the multiple packets of a single input ports arriving in the same time slot. In such cases, an arbitrary order is given to these packets by assigning different values to their extra precision digits. Because all the address cells of the same packet have the same timeStamp value, it can be used to identify the address cells that belong to the same multicast packet. The pDataCell field is a pointer to the data cell that the address cell corresponds to. When an address cell is scheduled to transfer, the input port will actually send the payloadData of the data cell that the address cells pDataCell field points to.

After explaining the two types of cells used, we now give the entire picture of the queue structure in a multicast VOQ switch. In each input port, there is a buffer used to store the data cells, and there are N virtual output queues to store the address cells for the N output ports. All the address cells in the same virtual queue are destined for the same output port, and only the address cells at the head of the queues can be scheduled. If an address cell receives the transmission grant from a particular output port in the scheduling, the crosspoint connecting the input port with the output port of the address cell will be set, and the data cell that the address cells pDataCell field points to will be transferred. After the data are sent, this address cell is removed from the head of its queue, and the fanoutCounter field of the corresponding data cell is accordingly decreased by one.

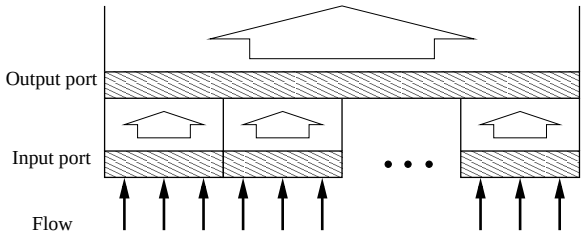


Fig. 2. The per flow fair scheduling for packet switches can be decomposed to two levels: the per port based fair scheduling and the per flow based fair scheduling.

IV. Ideal Model for Bandwidth Guaranteed Multicast Scheduling

In this section, we define an ideal model for bandwidth guaranteed multicast scheduling. The model fairly allocates the available bandwidth of an output port to all the input ports, and will be used as the reference system for our algorithm.

A. Per Port Scheduling and Per Flow Scheduling

A fair scheduling algorithm can provide fair bandwidth allocation at different granularity. We call it a per port scheduling algorithm if the input port is the unit of bandwidth allocation, and call it a per flow scheduling algorithm if the flow is the unit. For an efficient implementation, the per flow based fair scheduling for packet switches can be decomposed to two levels, as show in Fig.2. At the first level, per port fair scheduling algorithms on switches guarantee that the transmission capacity of each output port is fairly allocated to all the input ports. The first level enables each input port to get its reserved share of bandwidth from a specific output port. At the second level, the obtained bandwidth is further divided among different flows of this input port. Existing techniques for the second level of fair scheduling include fair scheduling algorithms for shared output links [9] - [15] and buffer management schemes [19] [20].

As seen in the previous section, the multicast VOQ switch uses the virtual output queued structure, and organizes address cells based on their destination output ports without distinguishing among flows. Nevertheless, the data cells in the buffer space still can be arranged on a per flow basis, and buffer management schemes similar to those in [19] [20] can be used to assure each flow guaranteed bandwidth. It is interesting and important to develop buffer management algorithms to guarantee the reserved bandwidths of multicast flows that share the same output links, but this is beyond the scope of this paper. In the following discussion, we focus on the first level of fair scheduling, i.e., to fairly assign the bandwidth of an output port to all the input ports.

B. Ideal Model for Per Port Multicast Fair Scheduling

For the convenience of establishing the model, we use the output queued switch as the underlying structure. When the input queued switch is considered, the per port fair scheduling must resolve two types of conflicts: (1) As in the usual scheduling, when multiple input ports have packets destined

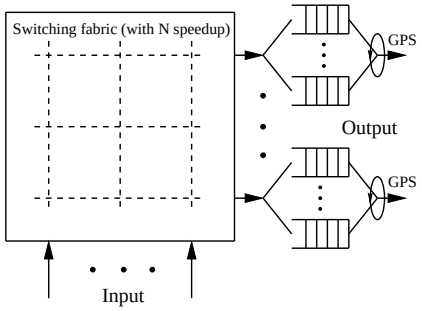


Fig. 3. The ideal model for per port multicast fair scheduling is based on the output queued switch, and each output port runs the GPS scheduler to ensure the fair bandwidth allocation.

to the same output port, only one can be granted to transmit at each time slot. (2) Further, for fairness guaranteed scheduling, the available bandwidth of an output port should be fairly divided among different input ports. On the input queued switch structure, it is difficult for a scheduling algorithm to fully satisfy both requirements at the same time. On the contrary, for the output queued switch, the first requirement is automatically satisfied, since with the N speedup, even each input port has a new incoming packet destined to the same output port, all of them can be transmitted through the switching fabric in the same time slot. Thus, a per port fair scheduling algorithm based on the output queued switch only needs to consider the bandwidth allocation issue.

Fig.3 shows the switch structure of the ideal model for per port multicast fair scheduling. It is an $N \times N$ output queued switch, and buffers the blocked packets at the output side using a per input port buffering strategy. In other words, each output port has N logic separate queues, so that packets arriving from different input ports can be placed in different queues. The crossbar switching fabric of the switch is capable of N speedup, and thus achieves 100% throughput. Upon the arrival of a unicast packet, it is immediately transmitted across the switch and delivered to its destination output port. For a multicast packet, the packet replication is done by the crossbar, and the packet is simultaneously sent to all its destinations.

An input port claims partial bandwidth on each output port as its reservation, and we denote the normalized (with respect to the total bandwidth of output port j) reserved bandwidth of input port i on output port j as r_{ij}. By the definition, $0 \leq r_{ij} \leq 1$, and to avoid overbooking at any input port or output port, r_{ij} satisfies that $\sum_{j=0}^{N-1} r_{ij} \leq 1$ for any $0 \leq i \leq N-1$, and $\sum_{j=0}^{N-1} r_{ij} \leq 1$ for any $0 \leq j \leq N-1$.

Each output port of the switch runs a GPS [8] scheduler to fairly allocate the available bandwidth to all the input ports according to their reservations. Equivalently, we can view each input port as having a logically separate and independent transmission channel at each output port. As a result, perfect fairness is achieved. Given that input port i_1 and i_2 have backlogged packets to output port j during the time interval $(t_1, t_2]$, the following equation always holds

$$\frac{B_{i_1 j}(t_1, t_2]}{B_{i_2 j}(t_1, t_2]} = \frac{r_{i_1 j}}{r_{i_2 j}}$$

where $B_{ij}(t_1, t_2]$ is the amount of bandwidth that input port i consumes on output port j in the interval $(t_1, t_2]$. In the next section, this model will be used by the CMF algorithm as the reference system to achieve fair bandwidth allocation.

V. Credit based Multicast Fair scheduling

In this section, we present the *Credit based Multicast Fair scheduling (CMF)* algorithm. CMF works on the multicast VOQ switch as described in Section III, and aims to efficiently schedule multicast traffic with bandwidth guarantees. The main idea for CMF to achieve fair bandwidth allocation is to track and minimize the difference between the bandwidth usage of an input port in the above ideal fairness model and that in the algorithm. On the other hand, CMF supports multicast scheduling by allowing a multicast packet to send transmission requests to multiple output ports simultaneously. Thus, the multicast packet has more chances to be delivered to all its destinations in the same time slot, and its multicast latency is shortened.

A. Terminologies

We introduce here some terminologies used to describe the CMF algorithm.

A *slot* is the unit of time for the switch to make scheduling decisions and transmit a batch of packets from input ports to output ports. Slots are numbered $0, 1, 2, \ldots$, and the switch starts to run at slot 0.

As in the ideal model, the *reservation* $r_{ij}(t)$ is the normalized reserved bandwidth of input port i on output port j at slot t. It is a function of the time slot index, because the reserved bandwidth may change at different time slots.

The *credit* $c_{ij}(t)$ is defined to be the usable bandwidth of input port i on output port j at slot t, i.e.,

$$c_{ij}(t) = \begin{cases} \frac{r_{ij}(t)}{\sum_{k \in I_j(t)} r_{kj}(t)}, & \text{if input } i \text{ has packets to output } j \text{ at slot } t \\ 0, & \text{otherwise} \end{cases}$$

where $I_j(t)$ is the set of input ports that have backlogged packets to output port j at slot t. In order to make full use of the available bandwidth, when an input port has no packet to send to a specific output port, its reserved bandwidth is reallocated to the rest backlogged input ports proportional to their reservations, and a GPS [8] scheduler handles the excessive bandwidth in the same way. Normally, $c_{ij}(t)$ does not need to be recomputed at each time slot, but instead only when the first new flow starts or the last existing flow ends.

The *balance* $b_{ij}(t)$ of input port i on output port j at slot t is the actual bandwidth it uses at this time slot. For an output port, either it is idle at a time slot, or one of the input ports is scheduled to send a packet through. In the latter case, the scheduled input port exclusively uses all the available bandwidth of the output port at this slot, and the rest of the input ports do not use any bandwidth, thus

$$b_{ij}(t) = \begin{cases} 1, & \text{if input } i \text{ sends a packet through output } j \text{ at slot } t \\ 0, & \text{otherwise} \end{cases}$$

Since CMF is a bandwidth guaranteed scheduling algorithm, we define the "accumulated credit" to record the up to date bandwidth usage.

The *accumulated credit* $A_{ij}(t)$ of input port i on output j till slot t is recursively defined as follows

$$A_{ij}(t) = \begin{cases} 0, & t = 0 \\ A_{ij}(t-1) + c_{ij}(t-1) - b_{ij}(t-1), & t \geq 1 \end{cases}$$

$A_{ij}(t)$ is the accumulated difference between the reserved bandwidth and the actually used bandwidth of input port i on output port j up to slot t. It is also the accumulated difference between the bandwidth that the input receives in the ideal fairness model and that in the algorithm, since in the ideal model, an input port gets exactly its reserved bandwidth. CMF achieves fairness bandwidth allocation by minimize the absolute value of the accumulated credit, and thus emulates the scheduling of the ideal fairness model.

We call $(A_{ij}(t) + c_{ij}(t))$ the available credit of input port i on output port j at slot t, which is the amount of bandwidth input port i can use at output port j at slot t without exceeding its reservation.

B. CMF Algorithm Description

Like most scheduling algorithms [3] [17] [18] [21] on VOQ switches, CMF is an iterative matching algorithm. An input port or an output port is said to be matched if it has been scheduled to send or receive a packet at the current time slot. Otherwise, it is free. Initially, all the input ports and output ports are free. After one iterative round finishes, some pairs of input ports and output ports get matched, and they will not be considered any more in the future rounds of the current time slot.

Each iterative round of CMF consists of two steps: (1) Request step. Address cells at each input port make requests to their destination output ports for possible transmission. (2) Grant step. Each output port selects one request from all the requests it received, and grants the transmission to the corresponding address cell. However, different from other three-step iterative algorithms, the accept step is not needed in CMF, because in our request step, all the address cells that make requests must point to the same data cell. Therefore, only one of the data cells in an input port can be granted the transmission, and there is no potential conflict in which an input port needs to send more than one data cells in a single time slot. In an iterative scheduling round, CMF has one fewer operational step, and less data exchange between input ports and output ports.

Next, we explain each step of CMF in more detail.

Before the scheduling starts, the accumulated credits of each input port are initialized to zero ($A_{ij}(0) = 0$), in the sense that no input port can pre-own credits.

Request Step. In the request step, if an input port is free, its earliest HOL address cells with positive available credits ($A_{ij}(t) + c_{ij}(t) > 0$) send requests to the corresponding output ports. There may be more than one such address cells in each input port, which come from the same multicast packet. Otherwise, if the input port has been matched with one or more output ports in this time slot, it means that a data cell has been scheduled to transmit, and therefore, no more address cells can make requests.

Giving priorities to the address cells with positive available credits helps CMF to achieve fair bandwidth allocation, i.e., firstly satisfying those that have not received enough bandwidth. Allowing the address cells of the same multicast to send requests simultaneously also gives the packet more chances to be transmitted to all its destinations in short latency.

Grant Step. After the request step, each output port has collected some requests. Like in the request step, requests with larger available credits will be given priorities, and each output port grants the request with the largest available credit.

Similarly, using the available credit as the grant criterion ensures fair bandwidth allocation. On the other hand, it also improves the chances that the address cells of the same multicast packet can simultaneously get grants from multiple output ports, because an input port normally claims reserved bandwidth based on its traffic flows, and thus has similar available credits on the multiple destination output ports of the same multicast flow.

The iterative rounds of the request and grant steps continue until no possible matching can be made.

However, at this time, there may still be matchable pairs of input ports and output ports, but are not matched because the HOL address cells have negative available credits and are masked out in the first stage of matching. Similar to WPIM [16], in order to improve the throughput of the algorithm and avoid wasting usable bandwidth, a second stage of matching is executed, which follows the same processes as in the first stage, except that the HOL address cells do not need positive available credits to send requests. The second stage matching will not affect the fairness properties of the algorithm, because the HOL address cells with positive available credits have been given priorities in the first stage, and those with negative available credits only consume the bandwidth that cannot be used by the former. Even the HOL address cells with negative available credit get scheduled, their accumulated credit will become smaller because of the newly generated balance, and their future chances of being transmitted are further reduced.

Data Transmission. After both stages of matching are completed, scheduling decisions are generated in the form of matched input port and output port pairs. Each input port usually has one data cell to send and may need to send this data cell to several output ports. On the other hand, each output port will receive no more than one data cell from an input port. The corresponding crosspoints connecting the scheduled input ports and output ports are set, and the input ports begin to send the data cells. Note that an input port may be connected to more than one output ports simultaneously. Thus, the algorithm can fully use the built-in multicast capability of the crossbar switching fabric.

Post Transmission Processing. When the crosspoints are set, all the input ports send their scheduled data cells to the

scheduled output ports at the same time. After the transmission is finished, the accumulated credits of each input port are updated accordingly, $A_{ij}(t+1) = A_{ij}(t) + c_{ij}(t) - b_{ij}(t)$. It may happen that although several input ports had backlogged packets to a specific output port, none of them obtained the chance to transfer due to other conflicts. In this case, the accumulated credits on this output port remain unchanged, i.e, $A_{ij}(t+1) = A_{ij}(t)$. Also, some post processing work needs to be performed to update the address cells and data cells that have been transferred. The served HOL address cells are removed from the heads of their queues, and the fanoutCounter fields of the related data cells are decreased accordingly. If a data cell's fanoutCounter field becomes 0, i.e., it has been sent to all destination output ports, the data cell is destroyed to return the buffer space.

VI. Simulation Results

We have conducted extensive simulations to compare the performance of CMF with other scheduling algorithms. The counterparts we compare CMF against include TATRA [6] and FIFOMS [7], which are multicast scheduling algorithms but do not provide bandwidth guarantees. By comparing with them, we show that CMF is indeed able to guarantee an input port its reserved bandwidth. The port scheduling versions of iFS [17] and iDRR [18] are also included in the simulations. These two algorithms are designed to fairly schedule unicast traffic on VOQ switches. By comparing with them, we demonstrate that CMF achieves short multicast latency.

Both pure unicast traffic and multicast traffic are adopted in the simulations. For a unicast packet, it has equal probability $(1/N)$ being destined to each output port. And a multicast packet has equal probability to go to any possible multicast destination. In other words, a multicast packet has the probability of 0.5 to be addressed to each output port. However, if a packet happens not to be addressed to any output port, it is regarded as invalid and discarded. Thus, the average fanout of a multicast packet is $0.5 \times N$.

We consider both Bernoulli arrivals and burst arrivals for unicast traffic and multicast traffic. The Bernoulli arrival is one of the most widely used models in the simulation of scheduling algorithms. Under the Bernoulli arrival, each input port has the probability of p to have a new packet to arrive at the beginning of a time slot. Therefore, the effective load is p for the Bernoulli unicast traffic and $0.5 \times N \times p$ for the Bernoulli multicast traffic.

In practice, network packets are usually highly correlated and tend to arrive in a burst mode. For a discrete time slot switch, we generally use a two state Markov process which alternates between off and on states to describe the burst nature. In the off state, there is no packet to arrive. In the on state, packets arrive at every time slot and all have the same destinations. At the end of each slot, the traffic can switch between off and on states independently. A burst traffic can be described using two parameters E_{off} and E_{on}. E_{off} is the average length of the off state, or alternatively the probability to switch from the off state to the on state is $1/E_{off}$. E_{on} is the average length of the on state, or the probability to switch from the on state to the off state is $1/E_{on}$. Therefore, the arrival rate is $E_{on}/(E_{off} + E_{on})$, and the effective load is $E_{on}/(E_{off} + E_{on})$ for the burst unicast traffic and $0.5 \times N \times E_{on}/(E_{off} + E_{on})$ for the burst multicast traffic. For easy comparison, we set E_{on} to be the same value 16 as in [6].

Each simulation runs for a fixed amount of time slots (10^6), and there is a sufficient warmup period (50% of the total simulation time) to obtain stable statistics.

In the following, we present the simulation results on different properties of the algorithms.

A. Bandwidth Guarantees

CMF minimizes the absolute value of the accumulated credit to assure the reserved bandwidth of each input port. By giving priorities to the address cells with more positive available credits in the scheduling, they are likely to be scheduled and have balances to reduce the accumulated credits. On the other hand, those with negative available credits are masked out from the first stage of matching, and have more chances to recover the accumulated credits by adding credits of the current time slot. The following results show that the fairness mechanism of CMF is effective.

A 4×4 switch is considered, with the following reservation setting:

$$\begin{pmatrix} r_{00} & r_{01} & r_{02} & r_{03} \\ r_{10} & r_{11} & r_{12} & r_{13} \\ r_{20} & r_{21} & r_{22} & r_{23} \\ r_{30} & r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.4 & 0.1 & 0.2 \\ 0.4 & 0.1 & 0.2 & 0.3 \end{pmatrix}$$

Ideally, input port 0, 1, 2, and 3 should receive 10%, 20%, 30%, and 40% bandwidth from output port 0, respectively. We let each input port have the same traffic load, and observe the actually obtained bandwidth of each input port. In the simulation, we assume there is limited buffer space at each input port and use a simple drop-tail buffer management strategy.

Fig.4 shows the actually received bandwidth of input port 0, 1, 2, and 3 on output port 0 in CMF. Initially, the load on each input port ($1/4$ of the effective load of the switch) is small, and all the arrived traffic can be totally delivered to the output port. As the load increases gradually, the switch can not sustain all the incoming traffic. The fairness mechanism becomes effective and prevents the input ports with small reservations from getting more than its reserved bandwidth. As a result, the actually obtained bandwidth of these input ports begin to drop. Finally, when the load on each input port goes beyond 40%, each input port can only get its reserved part of the bandwidth, which is 10%, 20%, 30%, and 40% respectively for input port 0, 1, 2, and 3. And the above observation holds for the Bernoulli multicast traffic (Fig.4(a)), burst multicast traffic (Fig.4(b)), Bernoulli unicast traffic (Fig.4(c)), and burst unicast traffic (Fig.4(d)).

Fig.5 describes the situation when FIFMOS is used. Since FIFOMS does not consider bandwidth guarantees, the total bandwidth is always equally allocated to all the input

ports. Fig.6 shows the results from TATRA. As can be seen, TATRA does not provide bandwidth guarantees either, and its maximum throughput is severely affected by the HOL blocking, especially under the burst unicast traffic (Fig.6(d)). It is interesting to note the small difference between the actually obtained bandwidth of each input port under the Bernoulli arrivals, as in Fig.6(a) and Fig.6(c), which can be explained by the fact that TATRA computes the "departure date" in an increasing order of input port indexes, and therefore the input ports with smaller indexes are given priorities in the scheduling.

B. Necessity of the Second Stage of Matching

In order to avoid wasting available bandwidth, CMF adds a second stage of matching to allow the address cells with negative available credit to be transmitted.

Fig.7 gives the actually obtained bandwidth of each input port with only the first stage of matching, under the same configuration as above. We can see that the bandwidth consumed by each input port is still roughly proportional to its reservation, which means the fairness mechanism is still effective. However, the available bandwidth of the output port is not guaranteed to be fully utilized, and each input port may get much less bandwidth comparing with the situations in Fig.4. The results show that the second stage of matching successfully increases the maximum throughput of the algorithm, and also does not affect the original fairness performance.

C. Multicast Latency

In order to show that CMF indeed supports the scheduling of multicast traffic, we compare the multicast latencies of various algorithms. The latency of a multicast packet is defined to be the time interval from the slot that the packet arrives at an input port to the slot that it is delivered to its last destination output port. Unicast here is viewed as a special case of multicast with fanout equal to 1.

To make the results more realistic, a 16×16 switch is considered. We assign each input port equal share of reserved bandwidth, i.e., $r_{ij} = 1/16$, and tested the average multicast latency. Fig.8(a) plots the multicast latency of the algorithms under Bernoulli multicast traffic. As can be seen, the three multicast scheduling algorithms, achieve much shorter latencies than the two unicast scheduling algorithms, which process a multicast packet as several copies of independent unicast packets. To be more specific, TATRA, FIFOMS, and CMF have almost the same latency when the load is not heavy, but the performance of TATRA drops dramatically when the effective load approaches 1 because of the HOL blocking. FIFOMS and CMF consistently give shorter latency than iFS and iDRR. Fig.8(b) shows the multicast latency of the algorithms under burst multicast traffic. Similar observations can be drawn that CMF and FIFOMS achieve shorter average multicast latency than iFS and iDRR. Note that since FIFOMS does not need to be concerned with the fairness property and works in a pure first-in-first-out manner, its delay under heavy load is smaller than that of CMF. Also, the HOL blocking makes TATRA have extremely large delay under heavy load. It also can be observed that, due to the bursty nature of the arrivals, the delay of any algorithm under the same effective load is much larger than that under the Bernoulli multicast traffic.

Fig.8(c) and Fig.8(d) show the results under Bernoulli unicast traffic and burst unicast traffic, respectively. Although specifically designed for multicast scheduling, CMF achieves short packet delay under pure unicast traffic, and successfully matches the two unicast scheduling algorithms, iFS and iDRR. Under unicast traffic, TATRA is more severely affected by the HOL blocking, and can only achieve a maximal throughput of about 55%, which is consistent with the theoretical analysis result of 58.6% in [25].

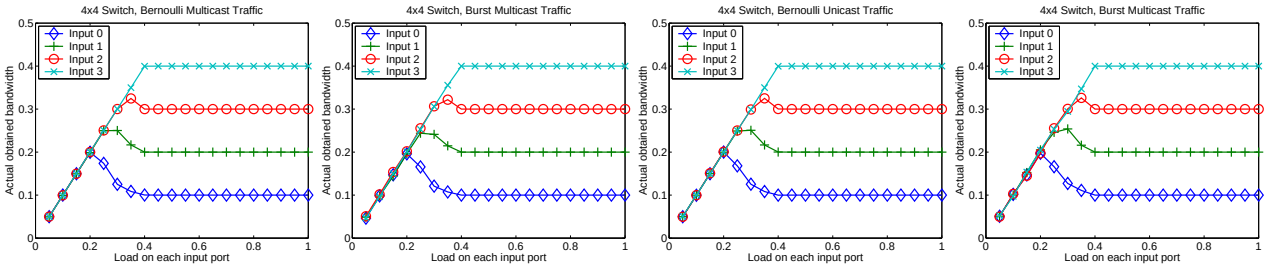
D. Convergence Rounds

Fig.9 compares the convergence rounds of the four iterative matching algorithms: CMF, FIFOMS, iDRR, and iWFQ. The same configuration is used as in testing multicast latency. We can see that the average convergence rounds of these algorithms are much smaller than N (=16). Under light load, the convergence rounds of all the algorithms are similar and not sensitive to the increase of the traffic. CMF has small convergence rounds under Bernoulli arrivals, but relatively large convergence rounds under burst arrivals. Generally, iDRR requires fewer rounds than others, because at the beginning of each time slot, the matched pairs of input ports and output ports can keep their matched status unless the assigned quota is used up.

VII. Conclusions

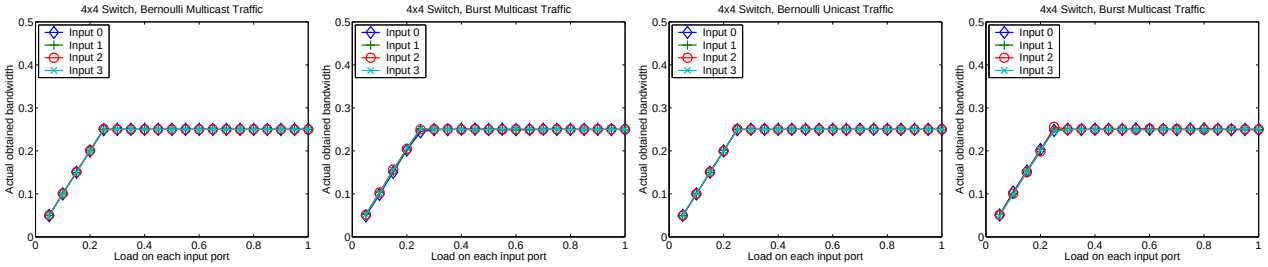
In this paper, we have proposed the Credit based Multicast Fair scheduling (CMF) algorithm to efficiently schedule multicast traffic with bandwidth guarantees. The multicast VOQ switch is adopted as the base of the algorithm. It stores the address information and the payload data of a packet separately, which allows an input port to manage only a linear number of queues for multicast traffic, and at the same time completely removes the HOL blocking.

CMF is an iterative matching algorithm, with each iterative round consisting of the request step and the grant step. CMF adopts a credit/balance based policy, and defines the accumulated credit to track the difference between the reserved bandwidth and the actually consumed bandwidth. It ensures the fair bandwidth allocation by minimizing the accumulated credit in the scheduling. At the same time, CMF supports multicast scheduling by allowing all the address cells of the same multicast packet to send transmission requests simultaneously, which increases the chance of this multicast packet being delivered to all its destinations in the same time slot, and thus shortens the multicast latency. Extensive simulations are conducted to evaluate the performances of CMF by comparing it against other algorithms. And the results demonstrate that CMF fulfills the design objectives:



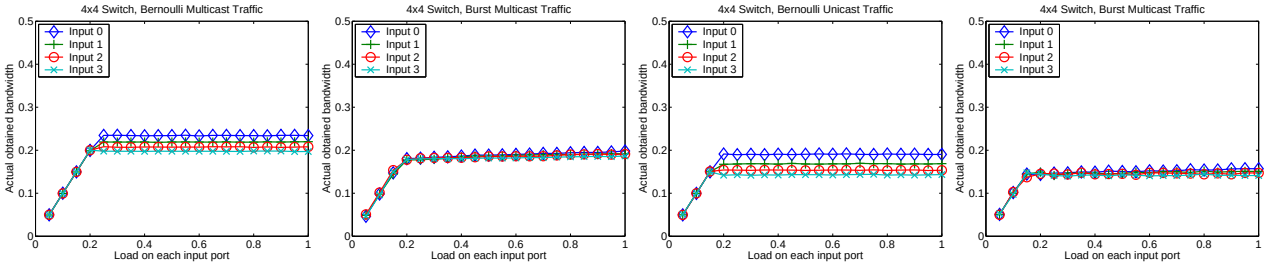
(a) (b) (c) (d)

Fig. 4. CMF provides bandwidth guarantee. (a) Actually obtained bandwidth under Bernoulli multicast traffic. (b) Actually obtained bandwidth under burst multicast traffic. (c) Actually obtained bandwidth under Bernoulli unicast traffic. (d) Actually obtained bandwidth under burst unicast traffic.



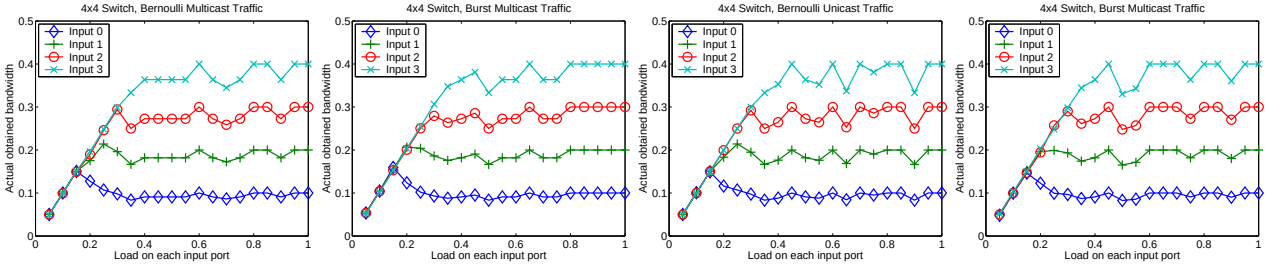
(a) (b) (c) (d)

Fig. 5. FIFOMS is not able to provide bandwidth guarantee. (a) Actually obtained bandwidth under Bernoulli multicast traffic. (b) Actually obtained bandwidth under burst multicast traffic. (c) Actually obtained bandwidth under Bernoulli unicast traffic. (d) Actually obtained bandwidth under burst unicast traffic.



(a) (b) (c) (d)

Fig. 6. TATRA is not able to provide bandwidth guarantee. (a) Actually obtained bandwidth under Bernoulli multicast traffic. (b) Actually obtained bandwidth under burst multicast traffic. (c) Actually obtained bandwidth under Bernoulli unicast traffic. (d) Actually obtained bandwidth under burst unicast traffic.



(a) (b) (c) (d)

Fig. 7. Without the second stage of matching, CMF cannot make fully use of the available bandwidth. (a) Actually obtained bandwidth under Bernoulli multicast traffic. (b) Actually obtained bandwidth under burst multicast traffic. (c) Actually obtained bandwidth under Bernoulli unicast traffic. (d) Actually obtained bandwidth under burst unicast traffic.

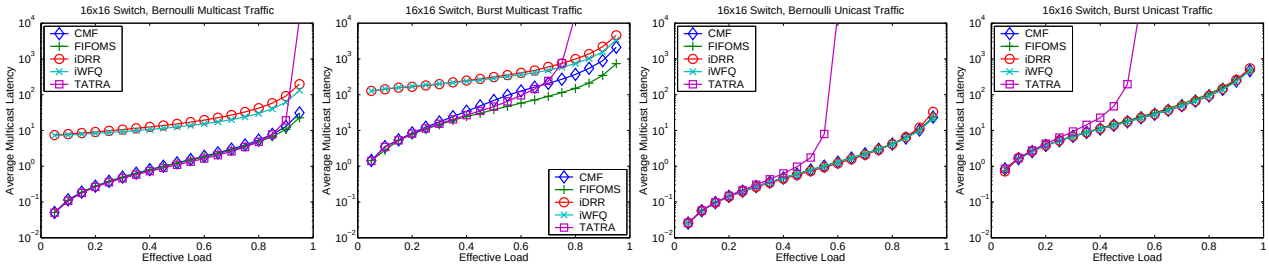


Fig. 8. Comparison of average multicast latency of different algorithms. (a) Average multicast latency under Bernoulli multicast traffic. (b) Average multicast latency under burst multicast traffic. (c) Average packet delay under Bernoulli unicast traffic. (d) Average packet delay under burst unicast traffic.

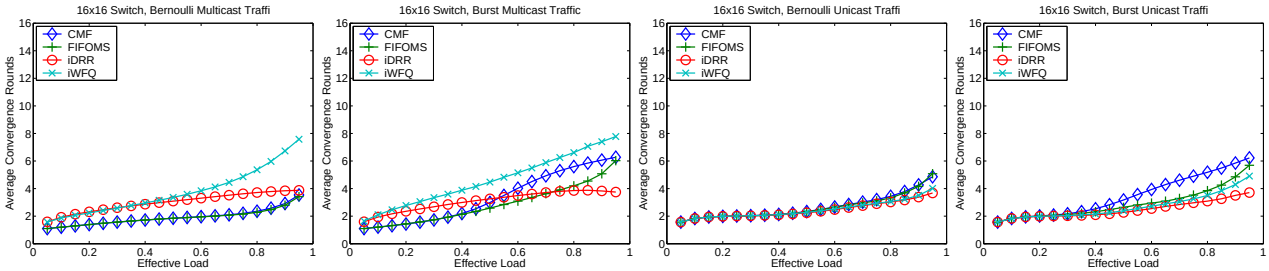


Fig. 9. Comparison of convergence rounds of the four iterative algorithms. (a) Average convergence rounds under Bernoulli multicast traffic. (b) Average convergence rounds under burst multicast traffic. (c) Average convergence rounds under Bernoulli unicast traffic. (d) Average convergence rounds under burst unicast traffic.

CMF is able to provide guaranteed bandwidth which most existing multicast algorithms, such as TATRA and FIFOMS, do not consider, and CMF outperforms the existing unicast fair scheduling algorithms, such as iFS and iDRR, by achieving short multicast latency.

REFERENCES

[1] H. Eriksson, "MBone: the multicast backbone," *Communications of the ACM*, vol. 37, no.8, pp. 54-60, Aug 1994.

[2] S. Keshav and Rosen Sharma, "Issues and trends in router design," *IEEE Communications Magazine*, vol. 36, no. 5, pp. 144-151, May 1998.

[3] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local-area networks," *ACM Transactions on Computer Systems*, vol. 11, no. 4, pp. 319-352, November 1993.

[4] S. Li, "Performance of a nonblocking space-division packet switch with correlated input traffic," *IEEE Trans. Commun.*, vol. 40, no. 1, pp. 97-108, Jan. 1992.

[5] B. Prabhakar, N. McKeown, and J. Mairesse, "Tetris models for multicast switches," *Proc. of the 30th Annual Conf. on Information Sciences and Systems*, Princeton, 1996.

[6] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling algorithms for input-queued switches," *IEEE J. Select. Areas Commun.*, vol. 15, no. 5, pp. 855-866, Jun. 1997.

[7] D. Pan and Y. Yang, "FIFO based multicast scheduling algorithm for VOQ packet switches," to appear in *IEEE Trans. Computers*, vol. 54, no. 10, October 2005.

[8] A. Parekh, R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344-357, Jun. 1993.

[9] A. Demers, S. Keshav, S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM '89*, vol. 19, no. 4, pp. 3-12, Austin, TX, Sep. 1989.

[10] J. Bennett, H. Zhang, "WF2Q: worst-case fair weighted fair queueing," *IEEE INFOCOM '96*, pp. 120-128, San Francisco, CA, Mar. 1996.

[11] M. Shreedhar, G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375-385, Jun. 1996.

[12] C. Guo, "SRR: an O(1) time complexity packet scheduler for flows in multi-service packet networks," *ACM SIGCOMM '01*, pp. 211-222, San Diego, CA, Aug. 2001.

[13] S. Cheung and C. Pencea, "BSFQ: bin sort fair queuing," *IEEE INFOCOM '02*, pp. 1640-1649, New York, Jun. 2002.

[14] S. Ramabhadran, J. Pasquale, "Stratified round robin: a low complexity packet scheduler with bandwidth fairness and bounded delay," *ACM SIGCOMM '03*, pp. 239-250, Karlsruhe, Germany, Aug. 2003.

[15] D. Pan and Y. Yang, "Credit based fair scheduling for packet switched networks," *Proc. of IEEE INFOCOM 2005*, Miami, FL, Mar. 2005

[16] D. Stiliadis and A. Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch," *Proc. IEEE INFOCOM '95*, pp. 960-968, Apr. 1995.

[17] N. Ni and L. Bhuyan, "Fair scheduling for input buffered switches," *Cluster Computing*, vol. 6, no. 2, pp. 105-114, Hingham, MA, Apr. 2003.

[18] X. Zhang, L. Bhuyan, "Deficit round-robin scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, no. 4, pp. 584-594, May 2003.

[19] S. Cheung and C. Pencea, "Pipelined sections: A new buffer management discipline for scalable QoS provision," IEEE Infocom '01, April 2001.

[20] R. Guerin, S. Kamat, V. Peris and R. Rajan, "Scalable QoS provision through buffer management," *ACM SIGCOMM 1998*, pp. 29-40, 1998.

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

[22] S.-T. Chuang, A. Goel, N. McKeown and B. Prabhkar, "Matching output queueing with a combined input output queued switch," *IEEE INFOCOM '99*, pp. 1169-1178, New York, March 1999.

[23] Nick McKeown, "A Fast Switched Backplane for a Gigabit Switched Router," *Business Communications Review*, vol. 27, no. 12, Dec. 1997.

[24] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellesick, and M. Horowitz, "The Tiny Tera: A packet switch core," *IEEE Micro*, vol. 17, no. 1, pp. 26-33, Feb. 1997.

[25] M. J. Karol, M. J. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347-1356, Dec. 1987.