Bandwidth Guaranteed Multicast Scheduling for Virtual Output Queued Packet Switches

Deng Pan
School of Computing & Information Sciences
Florida International University
Miami, FL 33199, USA

Yuanyuan Yang
Dept. of Electrical & 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 in the literature, they usually aim to achieve only short multicast latency and high throughput without considering 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 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, in order to fully utilize the multicast capability provided by the switch, CMF lets a multicast packet simultaneously send transmission requests to multiple output ports. In this way, a multicast packet has more chances to be delivered to multiple destination output ports in the same time slot and thus to achieve short multicast latency. Extensive simulations are conducted to evaluate the performance of CMF, and the results demonstrate that CMF achieves the two design

This research work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0207999 and CCF-0744234. A preliminary version of the paper appeared in IEEE BROADNETS 2005.

goals: fair bandwidth allocation and short multicast latency.

Keywords: Multicast, packet scheduling, fair scheduling, virtual output queued (VOQ) switch, quality of service (QoS).

I. Introduction

Multicast enables efficient data transmission from one source to multiple destinations. With the rapid development of broadband networks, multicast has been playing an important role in many Internet multimedia applications [1] [2], such as online gaming, video conferencing, and distance learning. Although a multicast packet can be handled as multiple copies of a unicast packet, it is desired that multicast scheduling and switching are supported in switches and routers to save network bandwidth and reduce multicast latency. In this paper, we consider multicast scheduling on packet switches. Such a switch can provide high speed interconnections among a group of processors in a parallel and distributed computing system. It can also be used as a crossconnect in an intermediate router or an edge router of a wide area communication network.

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. Thus 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 requirement 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 gigabits per second [3] [4].

On the contrary, an input queued switch stores blocked packets at the input side, and therefore eliminates the N speedup requirement. The single input queued switch, as shown in Fig. 1(b),

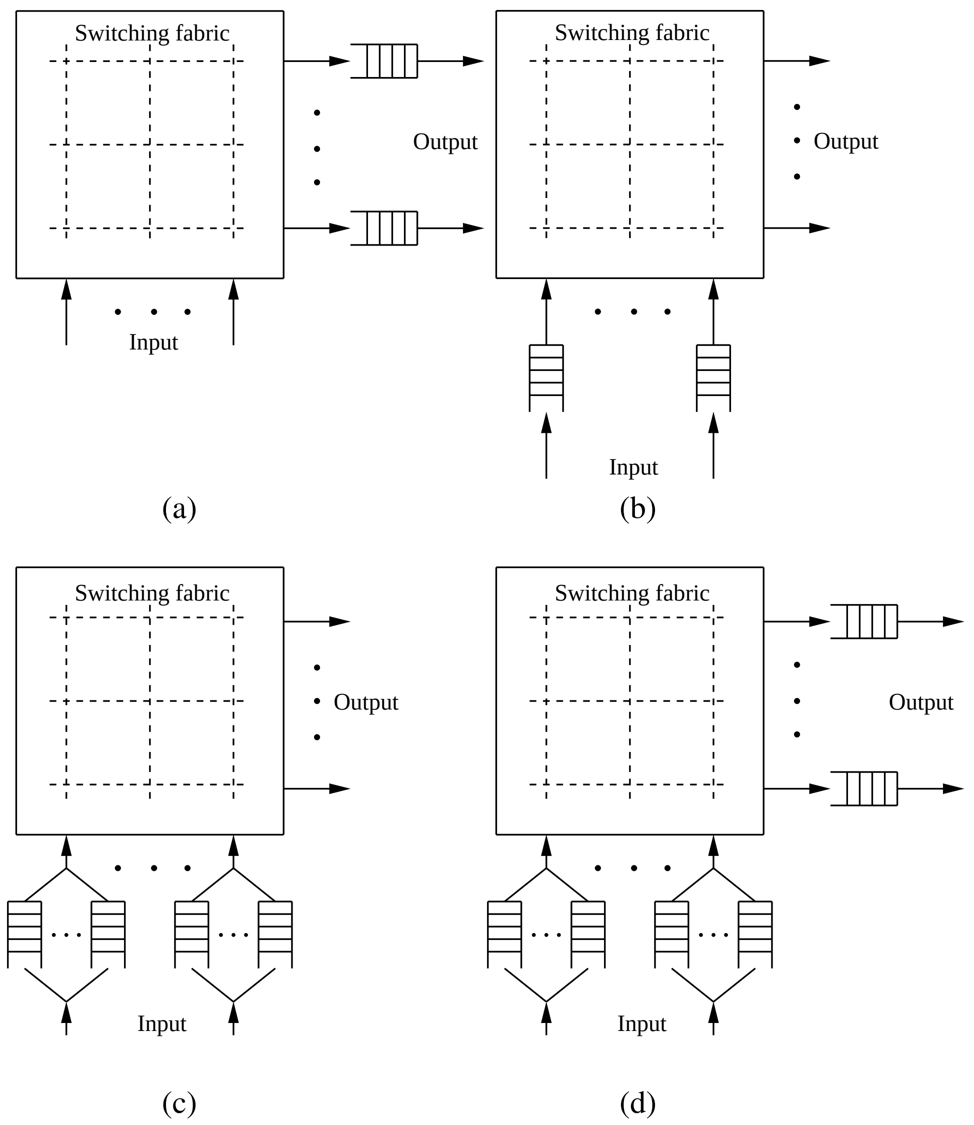


Fig. 1. Packet switches can be divided into different categories by the location where the blocked packets are buffered. (a) Output queued switch. (b) Single input queued switch. (c) Virtual output queued switch. (d) Combined input output queued switch.

has a first-in-first-out (FIFO) queue at each input port to store the incoming packets. Since only the packets at the head of line (HOL) of each 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 [5]. In order to eliminate the HOL blocking, the virtual output queued (VOQ) buffering was proposed [6]. A VOQ switch, as shown in Fig. 1(c), stores packets to different destinations in different logical queues. The HOL blocking is thus removed because a packet cannot be held up by another packet to a different destination. For unicast traffic, since a packet is destined for only one output port, there are only N possible destinations, i.e., the N different output ports, and therefore N logical queues are sufficient at each input port. However, the destination of a multicast packet can be any number of output ports, and thus there are $2^N - 1$ possible multicast destinations in total. For example, the multicast destinations of a switch with two output ports could be $\{out_0\}$, $\{out_1\}$ and $\{out_0, out_1\}$, where out_i denotes the i^{th} output port. As a result, $2^N - 1$ queues are necessary at each input port, which are not scalable. Clearly, a more efficient buffering strategy is needed to make the VOQ switch suitable for multicast traffic.

The combined input output queued (CIOQ) switch, shown in Fig. 1(d), extends the VOQ switch by adding speedup capability to the switching fabric. Thus, an output port may receive more than one packet in a single time slot, and needs buffer space to save the extra packets. It is proved in [7] that, for a CIOQ switch with speedup of two, 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 inexpensive hardware implementation, the input queued switch has been the main focus in the networking community, and several schemes have been proposed to schedule multicast traffic on the input queued switch, see, for example, TATRA [8], WBA [9], FIFOMS [10] and ESLIP [25]. However, existing multicast scheduling algorithms usually aim to achieve short delay and high throughput without fairness consideration. In order words, if malicious users inject an excessive amount of traffic in the network, those algorithms are not able to protect normal users from being affected. On the other hand, the quality of service (QoS) has become a major issue for the design of modern switches and routers, and fair bandwidth allocation is necessary in order to provide guaranteed performance services. Bandwidth guaranteed fair scheduling on shared output links has been well studied, and many algorithms have been proposed [12] - [18]. These algorithms can be easily applied to output queued switches to achieve fair bandwidth allocation, however, as mentioned earlier, output queued switches are expensive to implement due to the speedup requirement. Therefore, some efforts [19] [20] [21] have been made to apply these algorithms to input queued switches mainly for scheduling unicast traffic, and good 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 to simultaneously send a packet from one input port to multiple output ports in the same time slot. The new algorithm should be able to guarantee the reserved bandwidth of an input port at each output port and transmit multicast packets with short latency. As analyzed in [25], fixed length packet (also called cell) scheduling has significant advantages over variable length packet scheduling on VOQ switches, and is used by most of the implemented high speed VOQ switches, such as Cisco 12000 GSR [25], Tiny Tera [26] and AN2 [4]. When using such a switch in a network with variable length packets, such as the IP network, variable length packets are segmented to fixed length cells upon arrival at the input ports, and those cells are then used as the scheduling units and transmitted to the output ports, where they are reassembled back to the original packets before departure.

In this paper, we propose an algorithm called *Credit based Multicast Fair scheduling (CMF)*. CMF uses a credit 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 the difference to ensure fairness. Moreover, in order to fully utilize the multicast capability of the switch, CMF lets a multicast packet simultaneously send transmission requests to multiple output ports. In this way, a multicast packet has more chances to be delivered to multiple destination output ports in the same time slot and thus to achieve short multicast latency. We also conduct simulations under both multicast traffic and unicast traffic to evaluate the performance of CMF, and the results demonstrate that CMF fulfills the design objectives: fair bandwidth allocation and short multicast latency.

The rest of the paper is organized as follows. Section II reviews some existing schemes for bandwidth guaranteed fair scheduling and multicast 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 existing bandwidth guaranteed scheduling algorithms and multicast scheduling algorithms.

A. Bandwidth Guaranteed Scheduling on Shared Output Links

A lot of schemes have been proposed for bandwidth guaranteed fair scheduling on shared output links, such as 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* [12] and WF^2Q [13], 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 [11]. However, due to the operation to compare time stamps of the head packets of all the flows, time stamp based schedulers have high $O(\log M)$ time complexity, where M is the number of flows. (2) Round robin based. The scheduling principle of round robin schedulers, such as *DRR* [14] and *SRR* [15], is to serve the flows one by one, so that each flow has equal opportunity to transmit packets. 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 sending the next packet. (3) Combination of both. Some recently proposed algorithms, such as *BSFQ* [16] and *Stratified Round Robin* [17], 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* [19] improves upon *PIM* [4] 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 flows that have used up their quotas in the current frame are not allowed to participate in the scheduling for the remaining time slots of the frame. *iFS* [20] adapts *WFQ* [12], a time stamp based fair scheduler for shared output links, for VOQ switches. iFS uses a grant-accept two stage iterative matching method, and uses the virtual time as the grant criterion to emulate the GPS [11] ideal model at each output port. Similarly, *iDRR* [21] is the application of *DRR* [14], 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, it has a feature that a matched pair can keep the status until the assigned quota is used up, which 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 pay particular attention to multicast traffic, and thus they may not be able to achieve the best performance under mixed multicast/unicast traffic. *mFS* [20] 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 needs to maintain $2^N - 1$ separate queues at each input port in order to handle multicast traffic, which is impractical.

C. Multicast Scheduling Algorithms for Input Queued Switches

In general, a good multicast scheduling algorithm should be able to effectively utilize the multicast capability of the crossbar switching fabric. Existing multicast scheduling algorithms for input queued switches are usually designed to achieve short multicast latency and high throughput. TATRA [8] is a multicast scheduling algorithm for the single input queued switch. Its basic idea is to leave the residue, i.e., the set of packets that lose contention for output ports and remain at the HOL of the input queues, on the smallest number of input ports, so that more new packets can participate in the scheduling process of the next time slot. Because TATRA is difficult to implement in hardware and has high computational complexity, a cheaply implementable algorithm WBA [9] is proposed. WBA is a weight based algorithm, and cells with older ages or smaller fanouts are assigned more weights, for the purposes of fairness (starvation free) and residue concentration respectively. Once the weights are assigned, each output independently grants to the input with the highest weight. ESLIP [25] uses the VOQ structure to buffer unicast packets and puts all the multicast packets in a special single queue at each input port. It adopts a variant of the iSLIP [24] algorithm to schedule mixed unicast and multicast traffic. As can be expected, ESLIP eliminates the HOL blocking for unicast traffic, but not for multicast traffic. An iterative multicast scheduling algorithm FIFOMS for the multicast VOQ switch is proposed in [10]. Each iterative round of FIFOMS includes the request step and the grant step, and the first-in-first-out principle is used as the arbitration rule in both steps. FIFOMS demonstrates short multicast latency and small buffer space requirement in simulations.

III. Multicast VOQ Switch

In this section, we describe the multicast VOQ switch structure on which the CMF algorithm is based. 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 consists of 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 to which output port its packets should be sent. The second part of 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 needs to be sent only once from an input port to a single output port, it is natural to combine the two parts of information into a single unit and use it for both scheduling and data forwarding. 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 payload data. 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.

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

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 payload content of a packet. Since we assume that the switch operates on fixed length 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 the data cell can be destroyed to return the buffer space.

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 enters the switch, one address cell is created for each of its 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 port 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. For the timeStamp field size, 64 bits are sufficient (this is used by the time stamp counter in Pentium processors, and is good for a 100G bps ATM switch with fixed packet length of 53 bytes to run for more than 100 years without duplicating time stamps). Furthermore, by reasonably assuming the maximum lifetime of a packet in the switch, a finite length time stamp field can be used to represent unlimited time stamps using the techniques proposed in the literature [27] [28] [29]. 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. In the CMF algorithm, these address cells are allowed to simultaneously make transmission requests to the corresponding output ports, so as to increase the chances that a multicast packet is delivered to multiple destinations in the same time slot, and thus achieve short multicast latency.

The pDataCell field is a pointer to the data cell that the address cell corresponds to. Each address cell points to exactly one data cell, and a data cell may be pointed by one or more address cells due to the multicast traffic. When an address cell is scheduled to transfer, the input port will actually send the payloadData of the data cell that the pDataCell field of this address cell 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 to the same output port, and only the address cells at the head of the queues are eligible to 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 cell's 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.

In order to fit into the multicast VOQ switch queue structure, a packet needs to be processed upon arriving. One data cell and one or more address cells are generated and initialized as previously discussed. The data cell is put in the data buffer, and each address cell is placed at the end of the virtual queue based on its destination. It should be noted that the processing of new packets can be overlapped with the scheduling and switching of the switch, and thus it would not introduce extra delay for the packets.

Fig. 2 gives an example of a 4×4 multicast VOQ switch. The input ports and output ports are

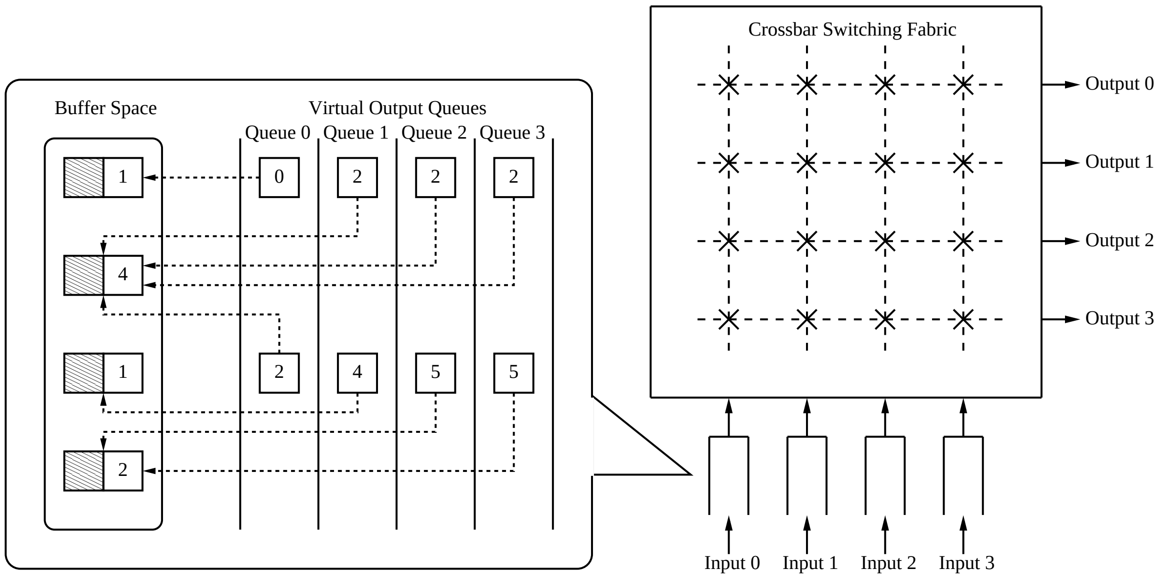


Fig. 2. An example of a 4×4 multicast VOQ switch. Left part shows the details of input port 0.

connected by a crossbar fabric, and incoming packets are buffered at the input side. The details of input port 0 are shown in the left part of the figure, in which there is a buffer for data cells and four virtual output queues for address cells. The shaded area of the data cell represents the payloadData field, and the number is the current value of the fanoutCounter. The number in the address cell stands for the timeStamp field, and the arrow points to its related data cell. Input port 0 has four packets that have not been fully transferred, and the packets entered the switch at the 0^{th}, 2^{nd}, 4^{th} and 5^{th} time slots, respectively. The current fanout of the 0^{th} slot packet is 1, and it still needs to be sent to output port 0. The 2^{nd} slot packet is destined to all the four output ports, and has not been transmitted to any one yet. The 4^{th} slot packet is a unicast packet to output port 1, and the destinations of the 5^{th} slot packet are output ports 2 and 3.

IV. Ideal Model for Bandwidth Guaranteed Multicast Scheduling

In this section, we present an ideal model for bandwidth guaranteed multicast scheduling. The ideal model allocates the available bandwidth of an output port to different input ports in a perfect fair manner. It is only used as a reference system by the CMF algorithm, and is not implemented in the simulations.

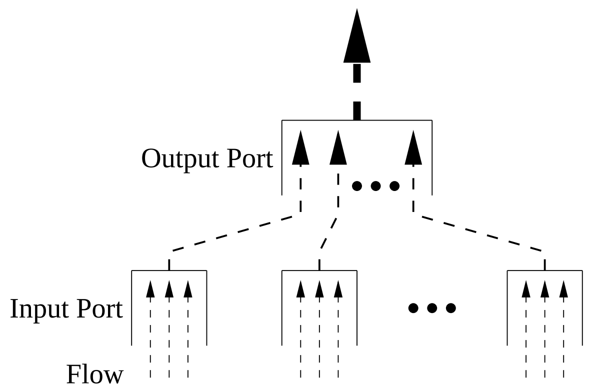


Fig. 3. The per flow fair scheduling for packet switches can be decomposed to two levels: the per port based fair scheduling and the per flow based fair scheduling.

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. 3. 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 obtain 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 [12] - [18] and buffer management schemes [22] [23]. In the following discussion, we focus on the first level of fair scheduling, i.e., fairly assigning the bandwidth of an output port to all the input ports.

B. Ideal Model for Per Port Multicast Fair Scheduling

We introduce an ideal model for per port multicast fair scheduling. The ideal model illustrates how perfect fairness can be achieved, and the CMF algorithm is designed to emulate the ideal model to provide similar bandwidth allocation. In other words, CMF tries to allocate to each input

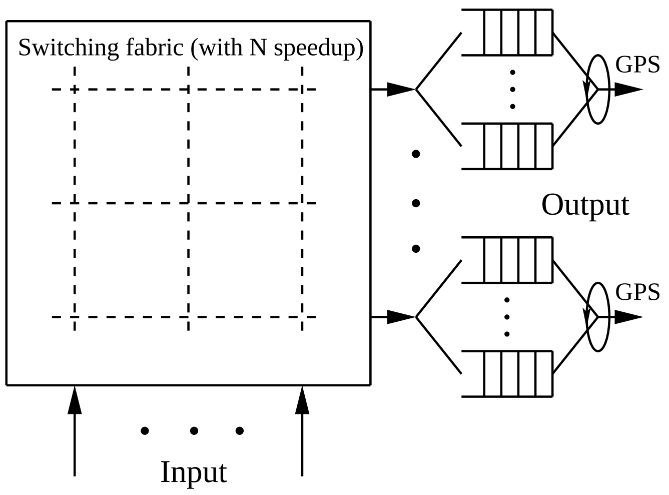


Fig. 4. The ideal model for per port multicast fair scheduling is based on the output queued switch, and each output port runs the GPS scheduler to ensure the fair bandwidth allocation.

port the same amount of bandwidth that it would receive in the ideal model.

For the convenience of establishing the model, we use the output queued switch as the underlying structure, because it would be very difficult if not impossible to build the ideal model based on the input queued switch. 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 to the same output port, only one can be granted to transmit at each time slot. (2) Furthermore, for fairness guaranteed scheduling, the available bandwidth of an output port should be fairly divided among different input ports. It is difficult for a scheduling algorithm of the input queued switch to resolve both types of conflicts at the same time. On the contrary, the output queued switch does not have the first type of conflict, 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. 4 shows the switch structure of the ideal model for per port multicast fair scheduling. It is an $N \times N$ output queued switch that buffers the blocked packets at the output side using a per input port buffering strategy. In other words, each output port has N separate logic 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 at time t as $r_{ij}(t)$. By the definition, $0 \leq r_{ij}(t) \leq 1$, and to avoid over-subscription at any input port or output port, $r_{ij}(t)$ satisfies that $\sum_{j=0}^{N-1} r_{ij}(t) \leq 1$ for any $0 \leq i \leq N-1$, and $\sum_{i=0}^{N-1} r_{ij}(t) \leq 1$ for any $0 \leq j \leq N-1$.

Each output port of the switch runs a GPS [11] scheduler, which serves the queues of different input ports in a weighted bit-by-bit round robin manner, and fairly allocates the available bandwidth to all the input ports according to their reservations. Equivalently, we can view it as that each input port has a logically separate and independent transmission channel at each output port. As a result, perfect fairness is achieved. Given that input ports i_1 and i_2 have backlogged packets to output port j during time interval (t_1, t_2), the following equation always holds

$$\frac{B_{i_1j}(t_1,t_2)}{B_{i_2j}(t_1,t_2)} = \frac{\int_{t_1}^{t_2} r_{i_1j}(t)dt}{\int_{t_1}^{t_2} r_{i_2j}(t)dt}$$

where $B_{ij}(t_1,t_2)$ is the amount of bandwidth that input port i consumes on output port j in interval (t_1,t_2). However, the relationship represented by the above equation between actually received bandwidth and reserved bandwidth is not explicit. On the other hand, when there is no new flow joining or existing flow leaving, the reserved bandwidth of each input-output pair keeps constant. Therefore, we can divide time into short intervals so that in each interval $r_{i_1j}(t)$ and $r_{i_2j}(t)$ have fixed values. Suppose that $r_{i_1j}(t)$ and $r_{i_2j}(t)$ are constant during interval (t_1,t_2), which we denote by r_{i_1j} and r_{i_2j}, the above equation can be simplified to

$$\frac{B_{i_1j}(t_1,t_2)}{B_{i_2j}(t_1,t_2)} = \frac{r_{i_1j}}{r_{i_2j}}$$

Now, the relationship between actually received bandwidth and reserved bandwidth becomes obvious: they should be proportional.

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.

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 fully 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 of backlogged input ports proportionally to their reservations, and a GPS [11] scheduler handles the excessive bandwidth in the same way. Normally, $c_{ij}(t)$ does not need to be recomputed at each time slot, instead, only when the corresponding virtual queue changes from empty to backlogged or from backlogged to empty.

The *balance* $b_{ij}(t)$ of input port i on output port j at slot t is the actual bandwidth that 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 minimizing 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 on output port j at slot t without exceeding its reservation.

B. CMF Algorithm Description

Like most scheduling algorithms [4] [20] [21] [24] 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 an iterative round, some pairs of input ports and output ports are matched, and they will no longer be considered 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 send requests to their destination output ports for possible transmissions. (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 [31], the accept step is not needed in CMF. As will be seen in the detailed description of the request step, only when multiple address cells of an input port are generated by the same multicast packet and point to the same data cell, they can send requests simultaneously. 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. Thus, 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 ($A_{ij}(0) = 0$), so 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 one 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., first satisfying those that have not received enough bandwidth. Allowing the address cells of the same multicast packet 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. As in the request step, requests with larger available credits will be given higher priorities, and each output port grants the request with the largest available credit.

Similar to the purpose of adding positive available credit restriction in the request step, 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 receive 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 found.

However, at this time, there may still be matchable input-output pairs that 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 [19], 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 process 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 are scheduled, their accumulated credit will become smaller because of the newly generated balance, and as a result, their future chances of being transmitted are further reduced.

Data Transmission. After both stages of matching are completed, scheduling decisions have been generated in the form of matched input-output 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 a specific 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 in a multicast switch, and thus the algorithm can fully utilize the built-in multicast capability of the crossbar switching fabric.

Post Transmission Processing. After the transmission is completed, 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 buffered packets to a specific output port, the output port received no packet, because all the input ports were matched with other output ports. 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 zero, i.e., it has been sent to all destination output ports, the data cell is destroyed to return the buffer space.

C. Complexity Analysis

Similar to most of iterative matching algorithms, CMF has theoretical time complexity of $O(N \log N)$. In the request step of one iteration, each input port spends $O(\log N)$ time to find from up to N time stamp values the smallest one, and let all HOL address cells with such a time stamp value send requests in parallel. In the grant step of one iteration, each output port spends $O(\log N)$ time to find from up to N requests the one with the largest available credit. Because in each iteration, at least one input-output pair will be matched, CMF converges in N iterations in the worst case. Thus, by combining the two parts, the time complexity of CMF is $O(N \log N)$.

However, in practice the average number of convergence iterations of CMF is usually much smaller than N, as will be seen from the simulation results in Section VI. This is because that it is highly possible that different input ports send requests to different output ports, and multiple input-output pairs are matched in one iteration, so that the algorithm can converge with less than N iterations. In fact, many iterative matching algorithms [4] [19] [24] use $O(\log N)$ [4] as an approximation of the average number of convergence iterations.

VI. Performance Evaluations

We have conducted extensive simulations to evaluate the performance of CMF and compare it with existing scheduling algorithms. Other algorithms considered in the simulations include TATRA [9] and FIFOMS [10], which are multicast scheduling algorithms targeting short multicast latency, and WPIM [19], iFS [20] and iDRR [21], which are unicast scheduling algorithms with bandwidth guarantees.

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, while 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. 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 [9].

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.

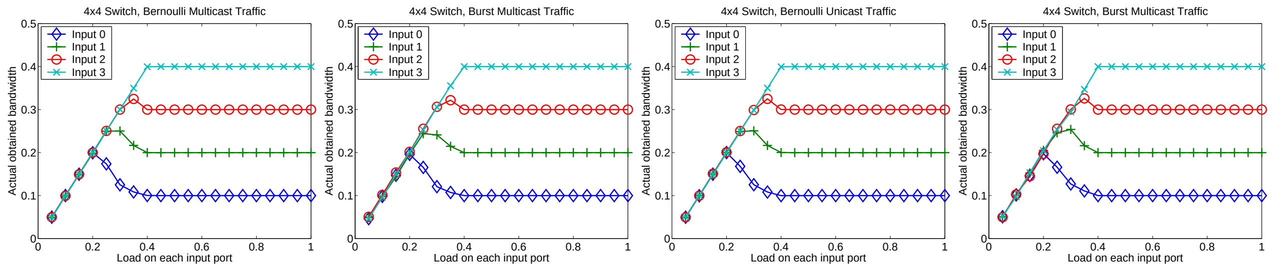
A. Bandwidth Guarantees

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 example reveals that the fairness mechanism of CMF is effective.

Consider a 4×4 switch 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 ports 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.

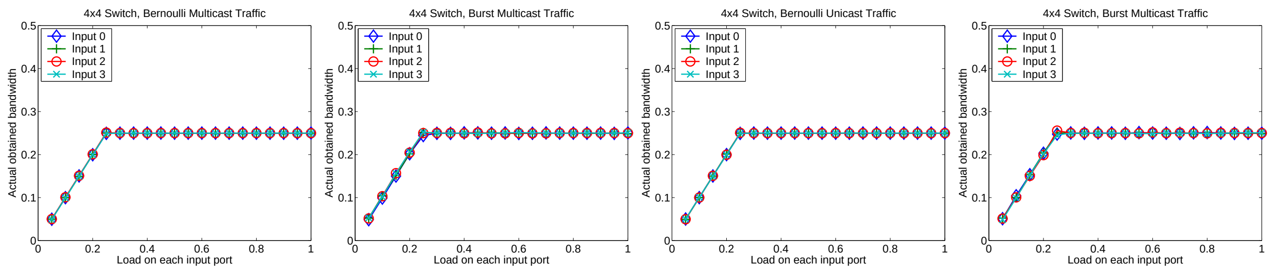


(a) (b) (c) (d)

Fig. 5. CMF provides bandwidth guarantees. (a) Under Bernoulli multicast traffic. (b) Under burst multicast traffic. (c) Under Bernoulli unicast traffic. (d) Under burst unicast traffic.

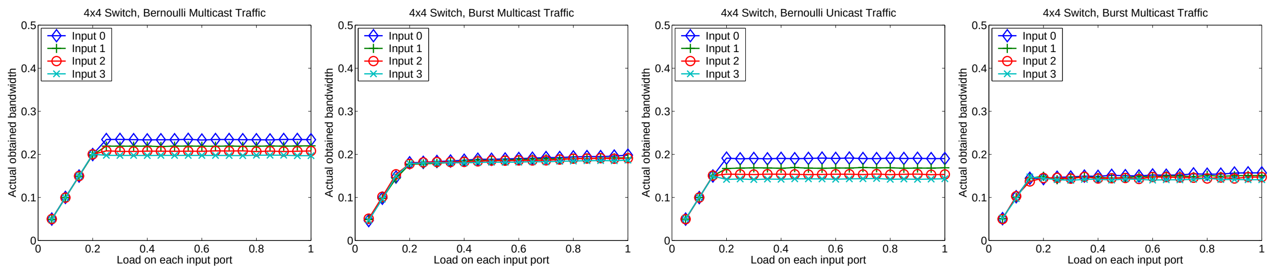
Fig. 5 shows the actually received bandwidth of input ports 0, 1, 2 and 3 on output port 0 in CMF. Initially, the load on each input port ($1/4$ of the effective load of the switch) is small, and all the arrived traffic can be completely delivered to the output port. As the load increases gradually, the switch cannot 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 begins 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 ports 0, 1, 2 and 3. The above observation holds for Bernoulli multicast traffic (Fig.5(a)), burst multicast traffic (Fig. 5(b)), Bernoulli unicast traffic (Fig. 5(c)) and burst unicast traffic (Fig. 5(d)).

Fig. 6 and Fig. 7 show the results of FIFOMS and TATRA, respectively. Because they do not take bandwidth guarantees into consideration, the total bandwidth is always equally allocated to all the input ports. As can be seen, the maximum throughput of TATRA is severely affected by the HOL blocking, especially under the burst unicast traffic (Fig. 7(d)). It is interesting to note that, when TATRA is used under Bernoulli arrivals, there is a small difference between the actually obtained bandwidth of different input ports, as shown in Fig. 7(a) and Fig. 7(c). This can be explained by the fact that TATRA computes the "departure date" in an increasing order of input port indexes, and hence the input ports with smaller indexes are given priorities in the scheduling.



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

Fig. 6. FIFOMS is not able to provide bandwidth guarantee. (a) Under Bernoulli multicast traffic. (b) Under burst multicast traffic. (c) Under Bernoulli unicast traffic. (d) Under burst unicast traffic.



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

Fig. 7. TATRA is not able to provide bandwidth guarantee. (a) Under Bernoulli multicast traffic. (b) Under burst multicast traffic. (c) Under Bernoulli unicast traffic. (d) Under burst unicast traffic.

B. Necessity of the Second Stage of Matching

As discussed earlier, 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. 8 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 that 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 receive much less bandwidth comparing with the situations in Fig. 5. The results show that the second stage of matching indeed increases the maximum throughput of the algorithm, and in the meanwhile does not affect the original fairness performance.

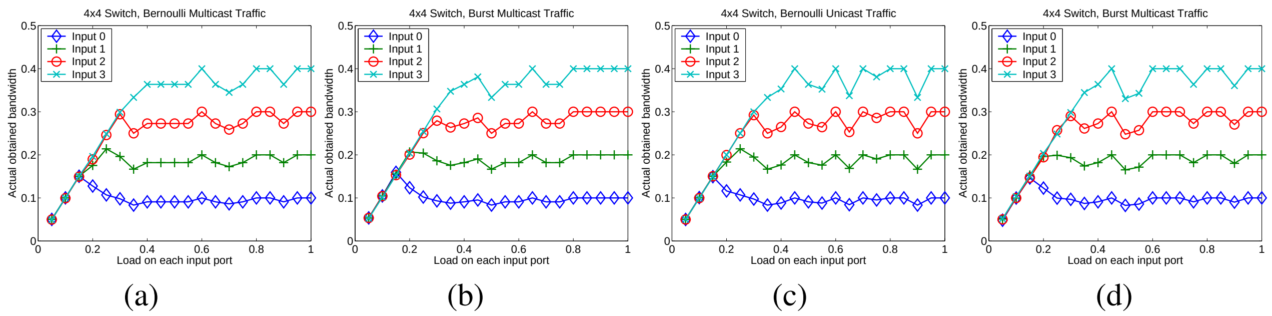


Fig. 8. Without the second stage of matching, CMF cannot make fully use of the available bandwidth. (a) Under Bernoulli multicast traffic. (b) Under burst multicast traffic. (c) Under Bernoulli unicast traffic. (d) Under burst unicast traffic.

C. Multicast Latency

We test whether the multicast scheduling mechanism of CMF is effective by evaluating its multicast latency. A multicast packet may be transmitted to its different destination output ports at different time slots, and multicast latency 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 is viewed as a special case of multicast with fanout equal to 1.

To obtain more realistic results, a 16×16 switch is considered in the rest of the simulations, and each input port is assigned equal share of reserved bandwidth, i.e., $r_{ij} = 1/16$. Fig. 9(a) plots the average multicast latency of different algorithms under Bernoulli multicast traffic. As can be seen, because the three unicast scheduling algorithms process a multicast packet as multiple copies of independent unicast packets, they have longer multicast latency than the three multicast scheduling algorithms. TATRA, FIFOMS and CMF have almost the same multicast latency when the load is not heavy, but the performance of TATRA drops dramatically when the effective load approaches 1 due to the HOL blocking. The simulation results under burst multicast traffic are given in Fig. 9(b). Similar observations can be drawn that CMF and FIFOMS achieve the shortest average multicast latency in most cases. 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 multicast latency under heavy load is shorter than that of CMF. It also can be noticed that, due to the bursty nature of the

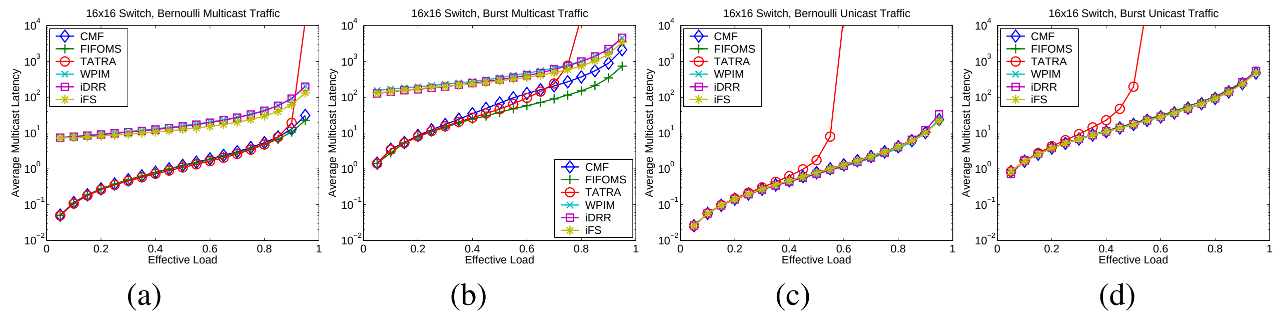


Fig. 9. Comparison of average multicast latency of different algorithms. (a) Under Bernoulli multicast traffic. (b) Under burst multicast traffic. (c) Under Bernoulli unicast traffic. (d) Under burst unicast traffic.

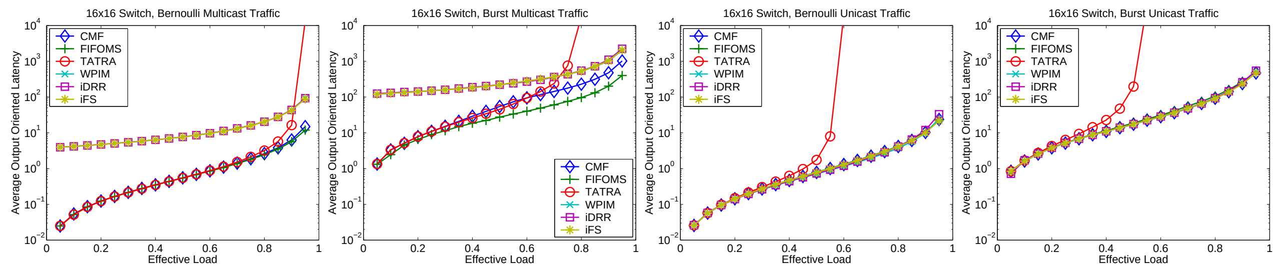
arrivals, the multicast latency of any algorithm under the same effective load is much longer than that under the Bernoulli multicast traffic, and TATRA saturates at about 80% effective load.

Fig. 9(c) and Fig. 9(d) show the results under Bernoulli unicast traffic and burst unicast traffic, respectively. We can see that although specifically designed for multicast scheduling, CMF performs equally well under pure unicast traffic, and successfully matches the unicast scheduling algorithms, WPIM, iFS and iDRR. Under unicast traffic, TATRA is more severely affected by the HOL blocking, and can only reach a maximal throughput of about 55%, which is consistent with the theoretical analysis result of 58.6% in [30].

D. Output Oriented Latency

Output oriented latency represents the transmission delay from the point of view of the receiver. It can be computed as the interval from the slot a packet enters the switch to the slot it is sent to the outline of one of its destinations. Thus, a multicast packet may be delivered to different destination with different output oriented latency.

Note that the output oriented latency performance criterion is not biased to multicast scheduling algorithms, because it does not consider the relationship between the different destinations of the same multicast packet, instead, it is only concerned with the arrival-to-departure interval. However, as shown in Fig. 10(a) and (b), the multicast scheduling algorithms still outperform the unicast scheduling algorithms, because they simultaneously consider the different destinations of



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

Fig. 10. Comparison of average output oriented latency of different algorithms. (a) Under Bernoulli multicast traffic. (b) Under burst multicast traffic. (c) Under Bernoulli unicast traffic. (d) Under burst unicast traffic.

a multicast during the scheduling. As can be expected, the average output oriented latency of each algorithm is shorter than its corresponding average multicast latency. Again, the output oriented latency of TATRA increases dramatically under heavy load. It is interesting to note that, although the three unicast scheduling algorithms use different scheduling policies, they have almost identical output oriented latency. The results under unicast traffic are shown in Fig. 10(c) and (d). As can be seen, under unicast traffic CMF achieves almost identical output oriented latency as the three unicast scheduling algorithms.

E. Convergence Rounds

Fig. 11 compares the convergence rounds of the five iterative matching algorithms: CMF, FIFOMS, WPIM, iDRR and iFS. Although in the worst case an iterative matching algorithm needs $N(=16)$ rounds to converge, we can see that the average convergence rounds of these algorithms are much smaller. 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.

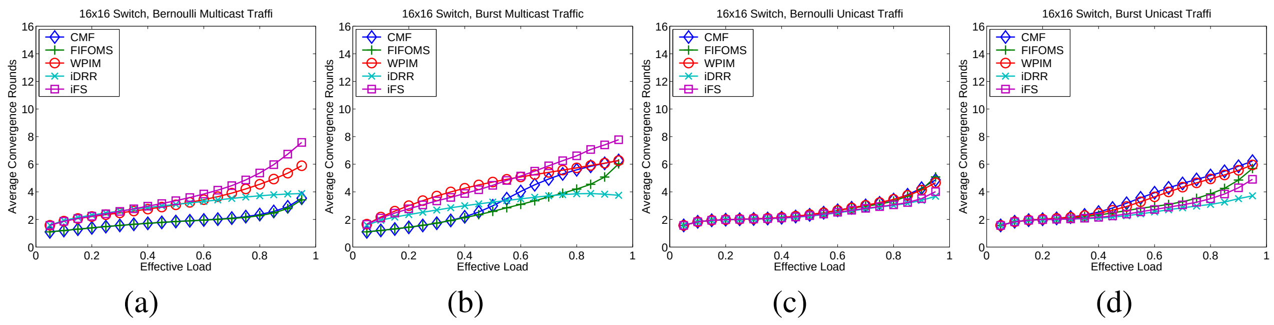


Fig. 11. Comparison of convergence rounds of the four iterative algorithms. (a) Under Bernoulli multicast traffic. (b) Under burst multicast traffic. (c) Under Bernoulli unicast traffic. (d) Under burst unicast traffic.

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 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 the multicast packet being delivered to multiple destinations in the same time slot, and thus achieves short multicast latency. Extensive simulations are conducted to evaluate the performances of CMF, and the results demonstrate that CMF fulfills the design objectives: fair bandwidth allocation and short multicast latency.

References

[1] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, “Revisiting IP multicast,” *ACM SIGCOMM 2006*, pp. 15-26, Pisa, Italy, Sep. 2006.

[2] L. Lao, J. Cui, M. Gerla, and D. Maggiorini, “A comparative study of multicast protocols: top, bottom, or in the middle?,” *IEEE INFOCOM 2005*, vol. 4, pp. 2809-2814, Miami, FL, Mar. 2005.

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

[4] T. Anderson, S. Owicki, J. Saxe and C. Thacker, “High speed switch scheduling for local-area networks,” *ACM Trans. Computer Systems*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[5] 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.

[6] Y. Tamir and G. Frazier, “High performance multi-queue buffers for VLSI communication switches,” *15th Annu. Symp. Comput. Arch.*, pp. 343-354, Jun. 1988.

[7] 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, Mar. 1999.

[8] B. Prabhakar, N. McKeown and J. Mairesse, “Tetris models for multi- cast switches,” *Proc. of the 30th Annual Conf. on Information Sciences and Systems*, Princeton, 1996.

[9] 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.

[10] D. Pan and Y. Yang, “FIFO based multicast scheduling algorithm for VOQ packet switches,” *IEEE Trans. Computers*, vol. 54, no. 10, pp. 1283-1297, Oct. 2005.

[11] A. Parekh and 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.

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

[13] J. Bennett and H. Zhang, “WF2Q: worst-case fair weighted fair queueing,” *IEEE INFOCOM ’96*, pp. 120-128, San Francisco, CA, Mar. 1996.

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

[15] 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.

[16] S. Cheung and C. Pencea, “BSFQ: bin sort fair queuing,” *IEEE INFOCOM ’02*, pp. 1640-1649, New York, Jun. 2002.

[17] S. Ramabhadran and 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.

[18] D. Pan and Y. Yang, “Credit based fair scheduling for packet switched networks,” *IEEE INFOCOM ’05*, pp. Miami, FL, Mar. 2005

[19] D. Stiliadis and A. Varma, “Providing bandwidth guarantees in an input-buffered crossbar switch,” *IEEE INFOCOM ’95*, pp. 960-968, Apr. 1995.

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

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

[22] S. Cheung and C. Pencea, “Pipelined sections: A new buffer management discipline for scalable QoS provision,” IEEE INFOCOM ’01, April 2001.

[23] R. Guerin, S. Kamat, V. Peris and R. Rajan, “Scalable QoS provision through buffer management,” *ACM SIGCOMM ’98*, pp. 29-40, 1998.

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

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

[26] 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.

[27] M. Mukund, K. Kumar, and M. Sohoni, “Bounded time-stamping in message-passing systems,” *Theoretical Computer Science*, vol. 290, no. 1, pp. 221-239, Jan. 2003.

[28] A. Mostefaoui and O. Theel, “Shrinking timestamp sizes of event ordering protocols,” *Proc. of 1998 International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 193-200, Dec. 1998.

[29] J. Renau, et al., “Tasking with out-of-order spawn in TLS chip multiprocessors: microarchitecture and compilation,” *ACM International Conference on Supercomputing (ICS)*, pp. 179-188, Jun. 2005.

[30] 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.

[31] D. Pan and Y. Yang, “Pipelined two step iterative matching algorithms for CIOQ crossbar switches,” *Proc. of*

2005 ACM Symposium on Architectures for Networking and Communications Systems (ANCS), pp. 41-50, Princeton, NJ, Oct. 2005.