Credit Based Fair Scheduling for Packet Switched Networks

Deng Pan
Dept. of Computer Science
Yuanyuan Yang
Dept. of Electrical & Computer Engineering
State University of New York, Stony Brook, NY 11794, USA

ABSTRACT

With the rapid development of Internet multimedia applications, the next generation of networks is required to schedule not only the best effort traffic but also the traffic with bandwidth and delay guarantees. Currently, there are two types of fair scheduling algorithms in the literature. The time stamp based schedulers achieve very good fairness and delay guarantees but have high $O(\log N)$ time complexity, where N is the number of flows. While the round robin based schedulers reach $O(1)$ time complexity, their delay guarantees are $O(N)$. This paper aims at a fair scheduling algorithm with constant time complexity as well as good fairness and delay guarantees. We first present a credit/balance based fair scheduling algorithm called Most Credit First (MCF). We theoretically prove that MCF can provide $O(\log N)$ fairness, delay and delay jitter guarantees, and demonstrate experimentally that it actually can achieve $O(1)$ guarantees. In order to reduce the $O(\log N)$ time complexity of MCF, we further present a more efficient variant of MCF, called Fast Most Credit First (FMCF). FMCF achieves $O(1)$ time complexity by utilizing approximation and synchronization, and at the same time preserves the $O(\log N)$ theoretical fairness, delay and delay jitter guarantees of MCF. We also implemented MCF and FMCF in NS2 simulator to compare the end to end delay performance with other fair scheduling algorithms. Our experimental results demonstrate that MCF outperforms two commonly used fair schedulers, and FMCF is able to closely match the performance of MCF with reduced time complexity.

Keywords: Scheduling, fair scheduling, time stamp schedulers, round robin schedulers, gateways, Generalized Processor Sharing (GPS).

I. INTRODUCTION

With the rapid development of Internet multimedia applications, the next generation of networks is required to provide services of different qualities for various types of traffic with different performance requirements. Network services can be broadly classified into two categories: guaranteed performance services and best effort services. For guaranteed performance services, resources are reserved for an allocated transmission rate, and the performance, such as bandwidth, delay and delay jitter, is bounded within prespecified ranges. Best effort services, as implied by the name, make use of the available transmission capacity and try the best to forward user traffic, but provide no service quality guarantee. The constant bit rate (CBR) and unspecified bit rate (UBR) services in ATM networks [1] belong to the guaranteed performance category and the best effort category, respectively.

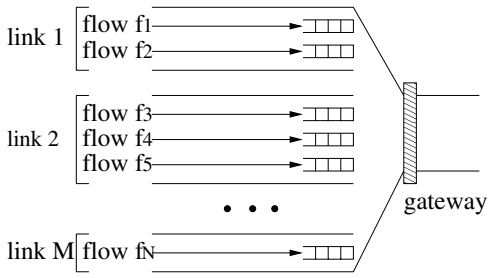


Fig. 1. A fair scheduler considers a gateway with incoming guaranteed performance flows and best effort flows, and schedules packets in a way that is able to ensure the requested bandwidths of guaranteed performance flows.

Efficiently scheduling guaranteed performance traffic and best effort traffic in the integrated service networks is an important and critical research issue, because the scheduling algorithms employed by the gateways/routers largely determine the service quality a network can provide. Ideal fair scheduling models, such as Generalized Processor Sharing (GPS) [3], usually consider a gateway (shared output link) with several incoming links, as shown in Fig. 1. Each link may include multiple guaranteed performance traffic flows or best effort traffic flows, and each flow has its (logically) independent queue to buffer the packets that have not been transmitted. The traditional FCFS algorithm schedules packets in the order of the packet arrival time, and is not able to protect a guaranteed performance flow from being affected by other ill-behaved flows. On the contrary, a fair scheduler can ensure a guaranteed performance flow the bandwidth it requests, and during any time interval the difference between the service a flow requests and it actually receives is bounded within a specified range, regardless of the length of the interval.

To design a good fair scheduling algorithm, one must take into consideration the following properties of the algorithm. 1) *Bandwidth guarantee* - The scheduler should limit each user to use only its share of bandwidth, so that ill-behaved users can be isolated from affecting normal users. 2) *Delay and delay jitter guarantees* - The scheduler should also make the bandwidth guarantee in an efficient way, so that the well-behaved users can have good and guaranteed delay and delay jitter performance. 3) *Low complexity* - In order to be applied to high speed backbone routers, the scheduler should have low time complexity, and in most cases, constant complexity is preferred so that the performance will not degrade as the number of users increases.

The fair scheduling problem has received a considerable amount of attention in the networking research communi-

0-7803-8968-9/05/$20.00 (c)2005 IEEE

ty, and several mechanisms have been proposed [3] - [13]. There are generally two types of fair schedulers: time stamp based and round robin based. Time stamp based schedulers can achieve good bandwidth and delay guarantees, but have $O(\log N)$ time complexity, where N is the number of flows. On the other hand, round robin based schedulers have $O(1)$ time complexity, but can provide only $O(N)$ delay guarantee.

In this paper we aim at designing a new fair scheduling algorithm with constant time complexity as well as good fairness and delay guarantees. We will first present a new fair scheduling algorithm, called *Most Credit First (MCF)*. Different from the two types of existing fair schedulers, MCF adopts a credit/balance based policy, and provides the bandwidth and delay guarantees by tracking and minimizing the difference between the service a flow should receive in the ideal fairness model and that it receives in the algorithm. We will theoretically prove that MCF can provide $O(\log N)$ fairness, delay and delay jitter guarantees, and demonstrate experimentally that it actually can achieve $O(1)$ guarantees. In order to reduce the $O(\log N)$ time complexity of MCF, we further present a more efficient variant of MCF, called *Fast Most Credit First (FMCF)*. FMCF achieves $O(1)$ time complexity by utilizing approximation and synchronization, and at the same time preserves the $O(\log N)$ theoretical fairness, delay and delay jitter guarantees of MCF. We also implemented MCF and FMCF in NS2 simulator to compare the end to end delay performance with other fair scheduling algorithms. Our experimental results demonstrate that MCF outperforms two commonly used fair schedulers, and FMCF is able to closely match the performance of MCF with reduced time complexity.

II. Related work

In this section, we briefly review some fair schedulers in the literature.

Time stamp based schedulers - A time stamp based scheduler computes a time stamp for each packet upon its arrival, and schedules packets in the order of the computed time stamps. Weighted Fair Queuing (WFQ) [3] is the first such time stamp based fair scheduler. It emulates the ideal Generalized Processor Sharing (GPS) [2] model by computing a starting service time flag and a finishing service time flag according to the scheduling effect of this packet in GPS, and transmitting packets in the increasing order of their finishing service time flags. Worst-case Fair Weighted Fair Queueing [4] [5] addresses the service discrepancy between WFQ and GPS, and achieves "worst case fairness". Similar to WFQ, Start-time Fair Queuing [6] associates a start tag and a finish tag with each packet, but schedules packets in the order of their start tags, and therefore improves the delay for low throughput flows. Other variants of WFQ include Self-Clocked Fair Scheduling [7] and Virtual Clock [8], which do not need to maintain a reference GPS server and hence can compute the time stamp in a more efficient way. Time stamp based fair schedulers are proved to have good fairness and delay guarantee [14] - [19]. However, because they need to sort packets in the order of their time stamps, they have at least $O(\log N)$ time complexity, which makes the time stamp based schedulers not suitable for links with many incoming flows or very high speed networks.

Round robin based schedulers - The fundamental scheduling principle of the round robin scheduler is to serve the flows one by one, so that each flow has equal opportunity being served. For example, in Deficit Round Robin (DRR) [9], each flow is assigned a quantum size proportional to its weight, and has a deficit counter to record the current unused portion of the allocated bandwidth. A backlogged flow is allowed to send packets up to the amount of the sum of its quantum and deficit counter. Once a flow is served, it needs to wait for other $N-1$ flows to be served before its next turn, which leads to a highly burst output for each flow. Smoothed Round Robin [10] and Uniform Round Robin [11] were proposed to improve the short term fairness property of DRR. Round robin based fair schedulers achieve $O(1)$ time complexity, but have poor delay bounds, usually proportional to N, as each flow has to wait for all other flows before transmitting the next packet.

Combination of both - Some recently proposed algorithms 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. Bin Sort Fair Queueing [12] puts each packet into one of the bins according to its virtual time stamp, and the packets in the same bin are not sorted and scheduled in a FIFO manner. Thus, each packet is scheduled approximately by its virtual time stamp with reduced sorting cost. Stratified Round Robin [13] groups flows into flow classes according to their weights, and uses the round robin approach for inter-class scheduling and the time stamp approach for intra-class scheduling. Although these schedulers improve the time complexity by reducing the number of items that need to be sorted, they still have $O(N)$ worst case delay due to the round robin nature.

III. An Ideal Fair Scheduling Model

Before presenting our algorithms, we first explain an ideal fair scheduling model in this section. As WFQ emulates GPS, our proposed MCF and FMCF algorithms make scheduling decisions according to this ideal fairness model.

The model considers a gateway (or a shared output link), as shown in Fig. 1, of bandwidth Φ with N backlogged incoming flows $F = \{f_1, \ldots, f_N\}$. Among them, one flow, say, f_1, is for best effort services, and others provide guaranteed performance services. (If there are more than one best effort flows, they can be grouped into one flow, and their packets can be buffered in the same queue.) Each guaranteed performance flow f_i ($2 \leq i \leq N$) reserves a portion of the gateway bandwidth ϕ_i ($0 < \phi_i \leq \Phi$), and all the best effort traffic packets are put in a FIFO queue of flow f_1 to make use of the bandwidth left by other flows. To avoid overbooking, admission control is adopted to make sure $\sum_{i=2}^{N} \phi_i \leq \Phi$. Equivalently, for easy representation, we can assign the leftover

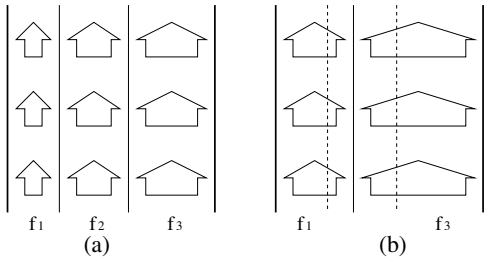


Fig. 2. Each flow has its logically independent transmission channel in the ideal fairness model. (a) Flow f_1, f_2 and f_3 have separate logical channels. (b) When flow f_2 is idle, its bandwidth is reallocated to f_1 and f_3 to make fully use of the available bandwidth of the gateway.

bandwidth $1 - \sum_{i=2}^{N} \phi_i$ to f_1 as its reserved bandwidth.

In this ideal fairness model, each flow logically has a separate and independent channel to transfer its packets as shown in Fig. 2(a), and hence it is straightforward to determine the transmission of the packets of each flow. For example, assume that all the packets have the same length l, and a new packet arrives at flow f_i at time t and is placed at the k^{th} position in the queue. Also assume that at time t, the head of line packet of f_i has not been served. Then the packet will begin to be served at time $t + \frac{(k-1)l}{\phi_i}$ and pass the gateway at time $t + \frac{kl}{\phi_i}$.

In the model, a flow is not allowed to accumulate bandwidth for future use. When a flow temporarily does not have any packet to transmit, its portion of the logical channel is idle. In such a case, in order to ensure high throughput and make fully use of the available bandwidth, the excessive idle bandwidth is reallocated according to some strategies, as shown in Fig. 2(b), such that $\sum_{i=1}^{N} \phi_i = \Phi$. The reallocation of the excessive bandwidth can be done in different ways for different purposes. For example, the excessive bandwidth can be allocated to each flow proportional to its existing reserved bandwidth. Or, the excessive bandwidth can be assigned to the best effort service flow to increase its transmission capacity when the reserved bandwidth of other flows is guaranteed.

IV. Most Credit First Fair Scheduling

In this section, we present the *Most Credit First* (MCF) fair scheduling algorithm. MCF achieves the bandwidth and delay guarantees by tracking and minimizing the difference between the bandwidth a flow would receive in the above ideal fairness model and that it actually uses in MCF.

A. Terminologies and Algorithm Description

In the following discussions, we assume a gateway (or shared output link) of R bandwidth with N flows $F = \{f_1, \ldots, f_N\}$. All the flow packets have the same fixed size, and the gateway runs in a time slotted manner. Similar to the ideal model, in MCF each flow claims a portion of the bandwidth as its reserved bandwidth by negotiating with the gateway before transmission. We first introduce some definitions and properties, and then describe the algorithm.

A *slot* is a unit of the time for one packet to pass through the gateway. Slots are numbered $0, 1, 2, \ldots$, and the gateway starts to run at slot 0.

The *reservation* $r_i(t)$ of flow f_i at slot t is the amount of bandwidth the flow reserves at this time slot. It is a function of the time slot index, because the reserved bandwidth of a flow may change at different time slots. The reservation of a flow satisfies that $0 < r_i(t) \leq R$.

For representational convenience, the total bandwidth of the gateway can be considered as one unit, and the reserved bandwidth of each flow is normalized as a ratio of its reservation to the total bandwidth of the gateway.

The *credit* $c_i(t)$ of flow f_i at slot t is the fraction of the output bandwidth flow f_i reserves at this time slot, i.e., $c_i(t) = \frac{r_i(t)}{R}$, and by the definition of the reservation, we have $0 < c_i(t) \leq 1$.

As in the ideal fairness model, if there is any excessive idle bandwidth, it is reallocated to avoid wasting available transmission capacity. Therefore, we have the following full bandwidth utilization property for normalized credits.

Property 1: If there is at least one backlogged flow, after the reallocation of the excessive idle bandwidth, the sum of the credits of all the flows is equal to unit, i.e.,

$$\sum_{i=1}^{N} c_i(t) = 1 \tag{1}$$

While "credit" stands for the reserved bandwidth of a flow, the term "balance" is borrowed to define the actual bandwidth a flow consumes at a given time slot.

The *balance* $b_i(t)$ of a flow f_i at slot t is the actual bandwidth it uses at this time slot. For a gateway, at each time slot, either it is idle, or one of the flows is scheduled to send a packet through the gateway. In the latter case, the scheduled flow exclusively uses all the available bandwidth at this slot, and the rest of the flows do not use any bandwidth, thus

$$b_i(t) = \begin{cases} 1, & \text{if flow } f_i \text{ is scheduled at slot } t \\ 0, & \text{otherwise} \end{cases}$$

Since MCF is a fairness oriented scheduling algorithm, we define the "accumulated credit" to record the up to date bandwidth usage of each flow.

The *accumulated credit* $A_i(t)$ of flow f_i till slot t is recursively defined as follows

$$A_i(t) = \begin{cases} 0, & t = 0 \\ A_i(t-1) + c_i(t-1) - b_i(t-1), & t \geq 1 \end{cases}$$

$A_i(t)$ is the accumulated difference between the reserved bandwidth and the actually used bandwidth of flow f_i up to slot t. It is initialized to 0 in the sense that no flow can pre-own credits before the beginning of the scheduling.

On the other hand, the "available credit" defines the usable credit of a flow at a given slot. It is used as the scheduling criterion in MCF.

The *available credit* $V_i(t)$ of flow f_i at slot t is the sum of its accumulated credit and the credit at this slot, i.e., $V_i(t) = A_i(t) + c_i(t)$.

We have the following property regarding the relationship between the accumulated credit and the available credit.

TABLE 1

MOST CREDIT FIRST FAIR SCHEDULING ALGORITHM

```
for each flow f_i do {
    initialize accumulated credit A_i = 0;
}

while true do {
    for each flow f_i do {
        compute available credit V_i = (A_i = A_i + c_i);
    }

    select a flow, say, f_k, with the largest available credit,
        V_k ≥ V_i for 1 ≤ i ≤ N;
    flow f_k sends a packet through the gateway;
    flow f_k updates its accumulated credit by A_k = A_k − 1;
}
```

Property 2: Based on the definitions of the accumulated credit and available credit, the following equations hold.

$$A_i(t) = \sum_{s=0}^{t-1} \Big(c_i(s) - b_i(s) \Big) = \sum_{s=0}^{t-1} c_i(s) - \sum_{s=0}^{t-1} b_i(s) \quad (2)$$

$$A_i(t+1) = V_i(t) - b_i(t) \quad (3)$$

MCF makes fairness and delay guarantees by restricting the absolute value of the accumulated credit. Its scheduling principle is to make the balances of a flow equal to its credits, in the sense that each flow consumes the same amount of bandwidth as it reserves. Therefore, MCF grants the flow with the most available credit to transmit, so that the flow can have "balance" and reduce its accumulated credit. On the other hand, the flows that use more bandwidth than they deserve in the previous slots and have negative available credits are penalized and not allowed to transmit in order to recover their accumulated credits.

The most credit first fair scheduling algorithm is formally described in Table 1. In the initialization stage, the accumulated credit of each flow is set to 0 before the scheduling. At the beginning of each time slot, each flow computes its available credit by adding its accumulated credit and the credit at this slot. Then, one flow is granted to transmit by using the available credit as the criterion. After the transmission, the accumulated credit of the scheduled flow is decreased by one. Since the reserved bandwidth of an idle flow is reallocated, the flow with the largest available credit, i.e. the scheduled flow, must be a backlogged flow. We will discuss the issue of reallocating excessive idle bandwidth in detail in Section VIII.

B. An Example of MCF Scheduling

To help understand how MCF works, we give a simple scheduling example here. There are three flows f_1, f_2, and f_3, with fixed credits $c_1(t) = c_1 = 0.1$, $c_2(t) = c_2 = 0.2$, and $c_3(t) = c_3 = 0.6$. Table 2 gives the scheduling decisions for the first eleven slots. The available credits in bold are the largest available credit of each time slot, and the corresponding flows get scheduled at each slot respectively. It can be seen that the slots any flow gets scheduled are roughly evenly distributed, e.g., slots 1, 5, and 8 for flow f_2. It is also interesting to note that after ten slots, each flow consumes the same amount of bandwidth as it reserves, and the accumulated credit of each flow goes back to zero, and thus fairness is achieved.

TABLE 2

A SCHEDULING EXAMPLE OF MCF

	f_1	f_2	f_3
c_i	0.1	0.3	0.6
$A_i(0)$	0.0	0.0	0.0
$V_i(0)$	0.1	0.3	**0.6**
$A_i(1)$	0.1	0.3	−0.4
$V_i(1)$	0.2	**0.6**	0.2
$A_i(2)$	0.2	−0.4	0.2
$V_i(2)$	0.3	−0.1	**0.8**
$A_i(3)$	0.3	−0.1	−0.2
$V_i(3)$	**0.4**	0.2	0.4
$A_i(4)$	−0.6	0.2	0.4
$V_i(4)$	−0.5	0.5	**1.0**
$A_i(5)$	−0.5	0.5	0.0
$V_i(5)$	−0.4	**0.8**	0.6
$A_i(6)$	−0.4	−0.2	0.6
$V_i(6)$	−0.3	0.1	**1.2**
$A_i(7)$	−0.3	0.1	0.2
$V_i(7)$	−0.2	0.4	**0.8**
$A_i(8)$	−0.2	0.4	−0.2
$V_i(8)$	−0.1	**0.7**	0.4
$A_i(9)$	−0.1	−0.3	0.4
$V_i(9)$	0.0	0.0	**1.0**
$A_i(10)$	0.0	0.0	0.0
$V_i(10)$	0.1	0.3	**0.6**

C. Bounds on Accumulated Credit

In order to see the fairness performance of MCF, we now derive bounds for the accumulated credit. From its definition, we know that the accumulated credit represents the difference between the service a flow requests and that it actually consumes. Thus, the range of its value is closely related to the fairness performance of MCF. First, we have the following lemma.

Lemma 1: In any single slot interval, the sum of the credits of all the flows is equal to the sum of the balances of all the flows, in the sense that the total bandwidth all the flows use in one time slot is equal to the total bandwidth all the flows are able to use, i.e.,

$$\sum_{i=1}^{N} c_i(t) = \sum_{i=1}^{N} b_i(t)$$

Proof. Consider two possible cases of the lemma.

Case 1: There is at least one backlogged flow. The full bandwidth utilization property (1) applies, $\sum_{i=1}^{N} c_i(t) = 1$. On the other hand, since the MCF is work conservative and there is a backlogged flow, the gateway should be busy at this slot, and one backlogged flow, say, f_k, is granted to transmit a packet. Thus, $\sum_{i=1}^{N} b_i(t) = b_k(t) = 1$.

Case 2: There is no backlogged flow. Because MCF does not allow flows to accumulate credits for future use, the credit of any flow should be 0, and $\sum_{i=1}^{N} c_i(t) = 0$. Also, since there is no backlogged flow, the gateway must be idle at the slot, and therefore $\sum_{i=1}^{N} b_i(t) = 0$.

Hence, in each case, we have $\sum_{i=1}^{N} c_i(t) = \sum_{i=1}^{N} b_i(t)$. ■

Theorem 1: The sum of the accumulated credits of all the flows at any time slot is 0, i.e.,

$$\sum_{i=1}^{N} A_i(t) = 0$$

Proof. By the definition of the accumulated credit, it records the difference between the bandwidth a flow is able to use and

that it actually uses. For all the N flows of the gateway, the sum of the difference at any time slot should be zero, which means that the actual bandwidth used by all the flows is always equal to the total available bandwidth.

From the non-recursive definition of the accumulated credit (3) of Property 2,

$$\sum_{i=1}^{N} A_i(t) = \sum_{i=1}^{N} \left(\sum_{s=0}^{t-1} \Big(c_i(s) - b_i(s) \Big) \right) = \sum_{s=0}^{t-1} \left(\sum_{i=1}^{N} c_i(s) - \sum_{i=1}^{N} b_i(s) \right)$$

And using Lemma 1, we have $\sum_{i=1}^{N} A_i(t) = 0$. ■

Corollary 1: If there is at least one backlogged flow, the sum of the available credits of all the flows is 1, i.e.,

$$\sum_{i=1}^{N} V_i(t) = 1$$

Proof. From the definition of the available credit, we have

$$\sum_{i=1}^{N} V_i(t) = \sum_{i=1}^{N} \Big(A_i(t) + c_i(t) \Big) = \sum_{i=1}^{N} A_i(t) + \sum_{i=1}^{N} c_i(t)$$

If there is at least one backlogged flow, the full bandwidth utilization property (1) applies. Combining this with Theorem 1, it follows that $\sum_{i=1}^{N} V_i(t) = 0 + 1 = 1$. ■

The following theorem shows that the accumulated credit of any flow in MCF is lower bounded by a constant.

Theorem 2: In MCF, the accumulated credit of any flow at any time slot is greater than or equal to $\frac{1}{N} - 1$, i.e.,

$$A_i(t) \geq \frac{1}{N} - 1 > -1$$

Proof. Since the accumulated credit of a flow only decreases after the flow gets scheduled, we only need to consider scheduled flows when computing the lower bound of the accumulated credit.

Suppose that flow f_k is scheduled at slot t, $b_k(t) = 1$. Thus, there is at least one backlogged flow at slot t and Corollary 1 applies. Furthermore, according to the scheduling policy, f_k has the largest available credit among all flows at slot t, and therefore $V_k(t) \geq \frac{1}{N}$. Using (3) of Property 2, we have,

$$A_k(t+1) = V_k(t) - b_k(t) = V_k(t) - 1 \geq \frac{1}{N} - 1 > -1$$

Thus, we have proved that the lower bound of the available credit is $\frac{1}{N} - 1$. ■

The next step is to derive the upper bound of the accumulated credit. In order to simplify the problem, in the rest of this section, we assume that each flow is always backlogged and its credit is kept as a constant $c_i(t) = c_i$. While the assumption does not weaken the generality of the results, it makes the analysis much easier.

We have an interesting observation from our simulation experiments (see Table 3 for an example) that the accumulated credit of any flow is always less than 2. In conjunction with the above proved lower bound $\frac{1}{N} - 1 (> -1)$, this reveals a nice property of MCF that the accumulated credit of any flow is in a constant range and is not sensitive to the increase of the number of flows. Accordingly, MCF should provide constant fairness and delay performance guarantees. However, it is difficult to formally prove this constant upper bound. A simple intuitive explanation is the follows. For a flow with small credit, once it is scheduled and has a balance, it needs a long time to recover its accumulated credit to be scheduled again, and therefore has a small chance to reach a very large value. For a flow with large credit, its available credit increases very quickly, and becomes the largest within a short period. Thus the flow has a big chance to be scheduled, and the corresponding large balances will prevent its accumulated credit from reaching too high. Although it is of course impossible to enumerate all the flow and credit configurations in MCF, we list in Table 3 several different cases to give some evidence of this constant upper bound. For the simulation configurations in the table, after the algorithm runs for a fixed number of slots, the accumulated credit of each flow goes back to zero, similar to the case in Table 2, because the total balances of each flow become equal to its total credits. Hence, the scheduling has a cycle, and it is possible to obtain the maximum value of the accumulated credit.

TABLE 3
EXAMPLES OF MCF FLOW AND CREDIT CONFIGURATIONS AND CORRESPONDING MAXIMUM ACCUMULATED CREDITS

N	Credit Configuration	Cycle	Max Ac. Credit
10	$c_{1..10} = 0.1$	10	0.9
10	$c_1 = 0.91, c_{2..10} = 0.01$	10^2	0.9
10	$c_1 = 0.21, c_2 = 0.31, c_3 = 0.41$ $c_{4..10} = 0.01$	10^2	1.12
10	$c_{1..2} = 0.46, c_{3..10} = 0.01$	10^2	1.18
10	$c_{1..3} = 0.31, c_{4..10} = 0.01$	10^2	1.32
10^2	$c_{1..10} = 0.091, c_{11..10^2} = 0.001$	10^3	1.629
10^2	$c_{1..20} = 0.046, c_{21..10^2} = 0.001$	10^3	1.628
10^3	$c_{1..10} = 0.0901, c_{11..10^3} = 0.0001$	10^4	1.728
10^3	$c_{1..30} = 0.0301, c_{31..10^3} = 0.0001$	10^4	1.826
10^4	$c_{1..100} = 0.00901, c_{101..10^4} = 0.00001$	10^5	1.879
10^4	$c_{1..200} = 0.00451, c_{151..10^4} = 0.00001$	10^5	1.879
10^5	$c_{1..1000} = 0.000901$ $c_{1001..10^5} = 0.000001$	10^6	1.889

Fortunately, at least we can theoretically derive a looser $O(\log N)$ upper bound for the accumulated credit. The basic idea is to assume that the maximum value of the accumulated credit is reached by one flow at a specific slot, and consider the sum of the accumulated credits of the flows that have been recently scheduled before that specific time slot. We trace back by including one more flow into consideration each time, until finally there is only one flow outside of the set of flows we are considering. Then, Theorem 2 can be applied to derive the bound of the maximum value we assumed.

First, we explain the notations to be used in the proof.

s: the time slot that the maximum accumulated credit reached.

M: the maximum value of the available credit.

$F(t, s)$: the set of flows that are scheduled in the inclusive time interval $[t, s]$ where $t \leq s$. Without loss of the generality, we assume that flow f_1 reaches the maximum accumulated credit M at slot s. Then, f_1 must be scheduled at slot s, otherwise $A_1(s+1) = A_1(s) + c_1 - b_1(s) = M + c_1 >$

M. Also assume that the sequence of the flows added into $F(t,s)$ when considering the sum of the accumulated credits is $f_1, f_2, \ldots, f_N$. In other words, when we look back from slot s, f_2 is the most recently scheduled flow. For example, if f_2 was most recently scheduled at slot $t_2(t_2 < s)$ before slot s, we have $F(s,s) = \{f_1\}$, $F(t_2+1,s) = \{f_1\}$, and $F(t_2,s) = \{f_1, f_2\}$. If f_3 was most recently scheduled at $t_3(t_3 < t_2)$, then $F(t_3+1,s) = \{f_1, f_2\}$ and $F(t_3,s) = \{f_1, f_2, f_3\}$.

$T(t,s)$: the sum of the accumulated credits of all the flows in $F(t,s)$ at slot t, i.e.,

$$T(t,s) = \sum_{f_i \in F(t,s)} A_i(t)$$

We next introduce some supporting lemmas needed in the proof of the accumulated credit upper bound.

Lemma 2: Suppose $F(t,s) = \{f_1, \ldots, f_k\}$ and $F(t-1,s) = \{f_1, \ldots, f_k, f_{k+1}\}$ for $k \geq 1$. Then,

$$T(t-1,s) \geq \frac{k+1}{k} T(t,s) - \sum_{i=1}^{k+1} c_i$$

Proof. By the definition of $F(t,s)$ and $F(t-1,s)$, flow f_{k+1} was scheduled at slot $t-1$. Since $V_{k+1}(t-1)$ was the largest among all available credits at slot $t-1$, it should be greater than or equal to the average of the rest, i.e.,

$$V_{k+1}(t-1) \geq \frac{1}{k} \sum_{i=1}^{k} V_i(t-1)$$

Moreover, since only f_{k+1} got scheduled, we have $b_{k+1}(t-1) = 1$ and $b_i(t-1) = 0$ for any $1 \leq i \leq k$, and therefore, by (3) of Property 2, $V_i(t-1) = A_i(t) - b_i(t-1) = A_i(t)$. Thus,

$$\sum_{i=1}^{k+1} V_i(t-1) = \sum_{i=1}^{k} V_i(t-1) + V_{k+1}(t-1)$$
$$\geq \frac{k+1}{k} \sum_{i=1}^{k} V_i(t-1) = \frac{k+1}{k} \sum_{i=1}^{k} A_i(t) = \frac{k+1}{k} T(t,s)$$

Also, by the definition of available credit, $A_i(t-1) = V_i(t-1) - c_i$ for any $1 \leq i \leq k+1$. Therefore,

$$T(t-1,s) = \sum_{i=1}^{k+1} A_i(t-1) = \sum_{i=1}^{k+1} \Big(V_i(t-1) - c_i \Big)$$
$$= \sum_{i=1}^{k+1} V_i(t-1) - \sum_{i=1}^{k+1} c_i \geq \frac{k+1}{k} T(t,s) - \sum_{i=1}^{k+1} c_i$$

■

Lemma 3: Suppose $F(t,s) = \{f_1, \ldots, f_k\}$ and $F(t-1,s) = F(t,s) = \{f_1, \ldots, f_k\}$ for $k \geq 1$. Then,

$$T(t-1,s) \geq T(t,s)$$

Proof. By the definition of $F(t,s)$ and $F(t-1,s)$, at slot $t-1$, the scheduled flow f_j belongs to the set $\{f_1, \ldots, f_k\}$, and therefore, $\sum_{i=1}^{k} b_i(t-1) = b_j(t-1) = 1$. By the recursive definition of the accumulated credit, we have

$$T(t,s) = \sum_{i=1}^{k} A_i(t) = \sum_{i=1}^{k} \Big(A_i(t-1) + c_i - b_i(t-1) \Big)$$
$$= \sum_{i=1}^{k} A_i(t-1) + \sum_{i=1}^{k} c_i - \sum_{i=1}^{k} b_i(t-1)$$
$$= T(t-1,s) + \sum_{i=1}^{k} c_i - 1$$

Since $\sum_{i=1}^{k} c_i \leq 1$ by the full bandwidth utilization property (1), we obtain $T(t-1,s) \geq T(t,s)$. ■

Lemma 4: Suppose $F(s,s) = \{f_1\}$ and $F(t,s) = \{f_1, \ldots, f_k\}$ for $k \geq 1$. Then

$$T(t,s) \geq k(M + c_1) - k \sum_{i=1}^{k} \left(c_i \sum_{j=i}^{k} \frac{1}{j} \right)$$

Proof. By induction on the number of flows in set $F(t,s)$,
Base case: $F(s,s) = \{f_1\}$, and $T(s,s) = A_1(s) = M \geq (M + c_1) - c_1$.
Inductive hypothesis: Suppose at slot t_1, $F(t_1,s) = \{f_1, \ldots, f_k\}$ and

$$T(t_1,s) \geq k(M + c_1) - k \sum_{i=1}^{k} \left(c_i \sum_{j=i}^{k} \frac{1}{j} \right)$$

We will prove that if at slot $t_3(t_3 < t_1)$, $F(t_3,s) = \{f_1, \ldots, f_k, f_{k+1}\}$, then

$$T(t_3,s) \geq (k+1)(M + c_1) - (k+1) \sum_{i=1}^{k+1} \left(c_i \sum_{j=i}^{k+1} \frac{1}{j} \right)$$

Assume that at slot $t_2(t_3 < t_2 \leq t_1)$, $F(t_2,s) = \{f_1, \ldots, f_k\}$, and $F(t_2-1,s) = \{f_1, \ldots, f_k, f_{k+1}\}$, i.e., flow f_{k+1} was scheduled or added into $F(t,s)$ at slot $t_2 - 1$. Then, by Lemma 3, $T(t_2,s) \geq T(t_1,s)$, and by Lemma 2,

$$T(t_2-1,s) \geq \frac{k+1}{k} T(t_2,s) - \sum_{i=1}^{k+1} c_i$$
$$\geq \frac{k+1}{k} T(t_1,s) - \sum_{i=1}^{k+1} c_i$$

Using the inductive hypothesis,

$$T(t_2-1,s)$$
$$\geq \frac{k+1}{k} \left(k(M + c_1) - k \sum_{i=1}^{k} \left(c_i \sum_{j=i}^{k} \frac{1}{j} \right) \right) - \sum_{i=1}^{k+1} c_i$$
$$= (k+1)(M + c_1) - (k+1) \sum_{i=1}^{k+1} \left(c_i \sum_{j=i}^{k+1} \frac{1}{j} \right)$$

Since $F(t_3,s) = F(t_2-1,s) = \{f_1, \ldots, f_k, f_{k+1}\}$, again by Lemma 3, we have

$$T(t_3,s) \geq T(t_2-1,s)$$
$$\geq (k+1)(M + c_1) - (k+1) \sum_{i=1}^{k+1} \left(c_i \sum_{j=i}^{k+1} \frac{1}{j} \right)$$

■

The upper bound on the accumulated credit is given in the following theorem.

Theorem 3: In MCF, the accumulated credit of any flow is less than $\ln N + C$, where C is a constant, i.e.,

$$A_i(t) \leq \ln N + C$$

Proof. Suppose at slot t, $F(t,s) = \{f_1, \ldots, f_{N-1}\}$, and at slot $t-1$, $F(t-1,s) = \{f_1, \ldots, f_N\}$. By Lemma 4,

$$T(t,s) \geq (N-1)(M+c_1) - (N-1)\sum_{i=1}^{N-1}\left(c_i \sum_{j=i}^{N-1}\frac{1}{j}\right)$$

On the one hand, since only f_N gets scheduled at slot t, $b_i(t-1) = 0$ for any $1 \leq i \leq N-1$, and by the recursive definition of the accumulated credit, $A_i(t-1) = A_i(t) - c_i + b_i(t-1) = A_i(t) - c_i$. Thus,

$$\sum_{i=1}^{N-1} A_i(t-1) = \sum_{i=1}^{N-1}\Big(A_i(t) - c_i\Big)$$

$$= \sum_{i=1}^{N-1} A_i(t) - \sum_{i=1}^{N-1} c_i = T(t,s) - \sum_{i=1}^{N-1} c_i$$

$$\geq (N-1)(M+c_1) - (N-1)\sum_{i=1}^{N-1}\left(c_i \sum_{j=i}^{N-1}\frac{1}{j}\right) - \sum_{i=1}^{N-1} c_i$$

On the other hand, since $V_N(t-1)$ is the largest available credit at slot t_1, from Corollary 1, $V_N(t-1) \geq \frac{1}{N}$, and by the definition of the available credit, $A_N(t-1) = V_N(t-1) - c_N \geq \frac{1}{N} - c_N$. Applying Theorem 1, we have

$$\sum_{i=1}^{N-1} A_i(t-1) = -A_N(t-1) \leq c_N - \frac{1}{N}$$

Combining the above two inequalities, we obtain

$$(N-1)(M+c_1) - (N-1)\sum_{i=1}^{N-1}\left(c_i \sum_{j=i}^{N-1}\frac{1}{j}\right) - \sum_{i=1}^{N-1} c_i$$

$$\leq c_N - \frac{1}{N}$$

and,

$$(N-1)(M+c_1) \leq \sum_{i=1}^{N} c_i - \frac{1}{N} + (N-1)\sum_{i=1}^{N-1}\left(c_i \sum_{j=i}^{N-1}\frac{1}{j}\right)$$

Because $\sum_{i=1}^{N} c_i \leq 1$, it follows that

$$M \leq \frac{1}{N} - c_1 + \sum_{i=1}^{N-1}\left(c_i \sum_{j=i}^{N-1}\frac{1}{j}\right)$$

$$< \frac{1}{N} - c_1 + \sum_{i=1}^{N-1}\left(c_i \sum_{j=1}^{N-1}\frac{1}{j}\right)$$

$$= \frac{1}{N} - c_1 + \left(\sum_{i=1}^{N-1} c_i\right)\left(\sum_{j=1}^{N-1}\frac{1}{j}\right) < \frac{1}{N} - c_1 + \sum_{j=1}^{N-1}\frac{1}{j}$$

$\sum_{j=1}^{N-1}\frac{1}{j}$ is the $(n-1)^{th}$ harmonic number [20], and

$$\sum_{j=1}^{N-1}\frac{1}{j} = \ln N + \gamma - \epsilon_N$$

where γ is the Euler's constant and there exists a constant K such that for a sufficiently large N, $|\epsilon_N| < \frac{K}{N}$.

Because both the values of $\frac{1}{N}$ and c_1 are bounded ($0 < \frac{1}{N} \leq 1$, $0 < c_1 \leq 1$), we can obtain,

$$M < \frac{1}{N} - c_1 + \sum_{j=1}^{N-1}\frac{1}{j} \leq \ln N + C$$

where C is a constant. ■

Now, we summarize the bounds for the available credit of MCF as follows:

Lower bound: $LB_{MCF} = \frac{1}{N} - 1$ (theoretically)
Upper bound: $UB_{MCF} = 2$ (experimentally), and
$= \ln N + C$ (theoretically)

It should be mentioned that although the theoretical $O(\log N)$ upper bound is good enough, as can be seen, it is quite loosely proved, and thus MCF can be reasonably expected to have better performance than the bound.

V. Fast Most Credit First Fair Scheduling

As mentioned in the introduction section, low time complexity is a requirement for a good fair scheduling algorithm. Although MCF exhibits good fairness guarantee by the tight bounds of the accumulated credit as shown in the last section, it still has the same time complexity as most time stamp based schedulers. That is, the time complexity of MCF is $O(\log N)$, where N is the number of flows, since it needs to find the flow with the largest available credit at each scheduling time slot. In order to reduce the time complexity of the algorithm and make it more practical to implement, in this section we further propose an efficient approximation of MCF, called *Fast Most Credit First (FMCF)*.

The basic idea of FMCF is to use approximation to achieve the simplicity of the operations. In FMCF, the scheduled flow does not need to have the largest available credit. On the contrary, its available credit could be g less than the largest available credit, where g is the granularity of the approximation. Its value affects the performance of FMCF, and usually a smaller value of g leads to a closer match to MCF. In order to achieve the approximation, $\lceil\frac{2+g}{g}\rceil$ "holes" are used in FMCF, as shown in Fig. 3. Each hole can hold only one flow whose available credit is in a specific range. Let $lastV$ denote the available credit of the scheduled flow at slot $t-1$. Then a flow with its available credit satisfying

$$lastV - 1 + (u-1)g < V_i(t) \leq lastV - 1 + ug$$

at slot t can be placed into the u^{th} hole.

Table 4 gives the pseudo code description of FMCF, and the details are explained as follows. At the beginning of each time slot, all the holes are set to empty (i.e., $hole[j] = -1$). After each flow computes its available credit, it tests if the hole ($hole[\lceil\frac{V_i - lastV + 1}{g}\rceil]$) corresponding to its available credit is free (i.e., $hole[\lceil\frac{V_i - lastV + 1}{g}\rceil] == -1$), and if yes, fills the hole with itself by setting $hole[\lceil\frac{V_i - lastV + 1}{g}\rceil] = i$. Theorem 4 assures that at least one hole must be filled at each slot. It also should be noted that it is possible that the available credits of several flows are in the range $(lastV - 1, lastV + 1 + g]$,

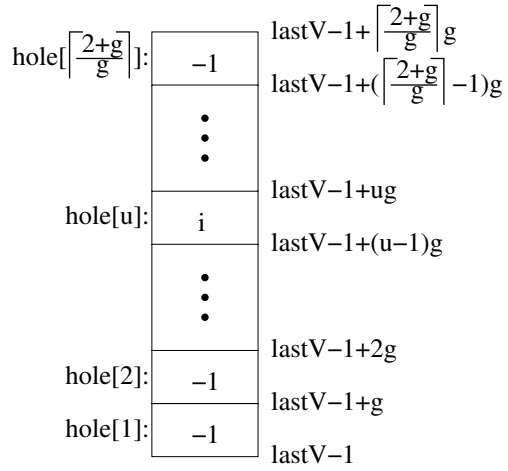


Fig. 3. $\lceil\frac{2+g}{g}\rceil$ "holes" are used in FMCF to achieve approximation. Each hole can hold only one flow whose available credit is in a specific range.

TABLE 4
FAST MOST CREDIT FIRST FAIR SCHEDULING ALGORITHM

```
for each flow f_i do {
    initialize accumulated credit A_i = 0;
}

lastV = 0;

while true do {
    for (j = 1; j ≤ ⌈(2+g)/g⌉); ++j) {
        hole[j] = −1;
    }

    for each flow f_i do {
        compute available credit V_i = (A_i = A_i + c_i);
        if (hole[⌈(V_i − lastV + 1)/g⌉] == −1)
            hole[⌈(V_i − lastV + 1)/g⌉] = i;
    }

    k = −1;
    for (j = 1; j ≤ ⌈(2+g)/g⌉); ++j) {
        if (hole[j] ≠ −1) k = hole[j];
    }

    flow f_k sends a packet through the gateway;
    lastV = V_k;
    flow f_k updates its accumulated credit by A_k = A_k − 1;
}
```

and there are more than one filled holes. The next step is to select the flow with the largest available credit from the filled holes, grant it to transmit a packet, and assign its available credit value to $lastV$ for the next slot scheduling. At the end, the accumulated credit of the scheduled flow is decreased by one as in MCF.

The correctness of FMCF is guaranteed by the following theorem.

Theorem 4: In FMCF, if the available credit of the scheduled flow at each time slot is at most g less than the maximum available credit at that slot, then the maximum available credit is in the range $(lastV - 1, lastV + 1 + g]$, where $lastV$ is the available credit of the flow scheduled at the previous time slot.

Proof. Suppose that at slot $t-1$, flow f_j has the largest available credit, $V_j(t-1) \geq V_i(t-1)$ for any $1 \leq i \leq N$, and f_k gets scheduled. By the definition of $lastV$, $lastV = V_k(t-1)$ for slot t, and $V_k(t-1)$ is at most g less than $V_j(t-1)$, i.e., $lastV + g \geq V_j(t-1)$.

Also, suppose that at slot t, flow f_l has the largest available credit, $V_l(t) \geq V_i(t)$ for any $1 \leq i \leq N$. Next, we prove that $lastV - 1 < V_l(t) \leq lastV + 1 + g$.

On the one hand, since $b_j(t-1) \leq 1$, we have

$$\begin{aligned} V_l(t) \geq V_j(t) &= V_j(t-1) - b_j(t-1) + c_j(t) \\ &\geq V_j(t-1) - 1 + c_j(t) \end{aligned}$$

And by $V_j(t-1) \geq V_k(t-1)$, it follows that

$$V_l(t) \geq V_k(t-1) - 1 + c_j(t) = lastV - 1 + c_j(t) > lastV - 1$$

On the other hand, since $b_l(t-1) \geq 0$, we can obtain

$$V_l(t) = V_l(t-1) - b_l(t-1) + c_l(t) \leq V_l(t-1) + c_l(t)$$

Using the fact that $V_l(t-1) \leq V_j(t-1)$ and $V_j(t-1) \leq lastV + g$, we have

$$V_l(t) \leq V_j(t-1) + c_l(t) \leq lastV + g + c_l(t) \leq lastV + g + 1$$

In addition, we need to show that the precondition of the theorem, that the available credit of the scheduled flow is at most g less than the maximum available credit, is always guaranteed by FMCF. Since the range of available credits of all the holes is $(lastV - 1, lastV - 1 + \lceil\frac{2+g}{g}\rceil g]$, and $lastV - 1 + \lceil\frac{2+g}{g}\rceil g \geq lastV + 1 + g$, from the above proof, we know that the available credit of f_l must be in the corresponding range of one of the holes, say, $hole[u]$. If $hole[u]$ is filled by f_l, there is no doubt that f_l is scheduled at slot t, because it has the largest available credit. Then the precondition holds. If $hole[u]$ is filled by another flow, say, f_m, then f_m is scheduled at slot t, because all the holes with greater corresponding ranges must be empty. Since f_m and f_l belong to the same hole, the difference of their available credits must be less than g. ■

The holes can be implemented by synchronization locks available in most operating systems. At the beginning of each slot, every lock is free. A flow tests if the lock is free before grabs it. If the lock is free, the flow sets the lock, and thereafter this lock is no longer available to other flows. Due to the approximation mechanism and the synchronization locks, FMCF only needs to compare at most $\lceil\frac{2+g}{g}\rceil$ flows in the filled holes, and $\lceil\frac{2+g}{g}\rceil$ is a constant when g is fixed. Therefore, the time complexity of FMCF is reduced to $O(1)$.

Note that although with reduced complexity, FMCF still has an equivalent available credit range as MCF:

$$\begin{aligned} &\textit{Lower bound}: LB_{FMCF} = \tfrac{1}{N} - g - 1. \\ &\textit{Upper bound}: UB_{FMCF} = (1+g)\ln N + C' \end{aligned}$$

which we summarize into the following theorems.

Theorem 5: In FMCF, the accumulated credit of any flow at any time slot is greater than or equal to $\frac{1}{N} - 1 - g$, i.e.,

$$A_i(t) \geq \frac{1}{N} - g - 1$$

Proof. It is easy to see that although the scheduling policy has been changed, Theorem 1 and Corollary 1 still apply to FMCF. The proof is similar to that of Theorem 2. Suppose that at slot t, flow f_j has the largest available credit and flow

f_k is scheduled. We have $f_j \geq \frac{1}{N}$ by Corollary 1, $b_k(t) = 1$ by the definition of balance, and $V_k(t) \geq V_j(t) - g$ by the scheduling principle of FMCF. Thus,

$$\begin{aligned} A_k(t+1) &= V_k(t) - b_k(t) = V_k(t) - 1 \\ &\geq V_j(t) - g - 1 \geq \frac{1}{N} - g - 1 \end{aligned}$$

■

Theorem 6: In FMCF, the accumulated credit of any flow at any time slot is less than $(1+g)\ln N + C'$, where C' is a constant, i.e.,

$$A_i(t) < (1+g)\ln N + C'$$

To prove Theorem 6, we need the following variants of Lemma 2 and Lemma 4 (Lemma 3 still holds for FMCF). All the variables used below have the same meaning as in Section IV.

Lemma 5: Suppose $F(t,s) = \{f_1, \ldots, f_k\}$ and $F(t-1,s) = \{f_1, \ldots, f_k, f_{k+1}\}$ for $k \geq 1$. Then,

$$T(t-1,s) \geq \frac{k+1}{k} T(t,s) - \sum_{i=1}^{k+1} c_i - g$$

The proof is similar to that of Lemma 2 and omitted.

Lemma 6: Suppose $F(s,s) = \{f_1\}$ and $F(t,s) = \{f_1, \ldots, f_k\}$ for $k \geq 1$. Then,

$$T(t,s) \geq k(M + c_1 + g) - k\sum_{i=1}^{k}\left(c_i \sum_{j=i}^{k} \frac{1}{j}\right) - kg\sum_{j=1}^{k}\frac{1}{j}$$

The proof is similar to that of Lemma 4 and omitted.

We now prove Theorem 6.

Proof. Suppose that at slot t, $F(t,s) = \{f_1, \ldots, f_{N-1}\}$, and at slot $t-1$, $F(t-1,s) = \{f_1, \ldots, f_N\}$. By Lemma 6,

$$\begin{aligned} T(t,s) \geq\ & (N-1)(M + c_1 + g) \\ & - (N-1)\sum_{i=1}^{N-1}\left(c_i \sum_{j=i}^{N-1} \frac{1}{j}\right) - (N-1)g\sum_{j=1}^{N-1}\frac{1}{j} \end{aligned}$$

and,

$$\begin{aligned} \sum_{i=1}^{N-1} A_i(t-1) &= T(t,s) - \sum_{i=1}^{N-1} c_i \\ &\geq (N-1)(M + c_1 + g) - (N-1)\sum_{i=1}^{N-1}\left(c_i \sum_{j=i}^{N-1} \frac{1}{j}\right) \\ &\quad - (N-1)g\sum_{j=1}^{N-1}\frac{1}{j} - \sum_{i=1}^{N-1} c_i \end{aligned}$$

Also, since $V_N(t-1) \geq \frac{1}{N} - g$, by Theorem 1,

$$\sum_{i=1}^{N-1} A_i(t-1) = -A_N(t-1) \leq c_N - (\frac{1}{N} - g)$$

Combining the above two inequalities and using the full bandwidth utilization property (1), we obtain

$$\begin{aligned} & (N-1)(M + c_1 + g) - (N-1)\sum_{i=1}^{N-1}(c_i \sum_{j=i}^{N-1}\frac{1}{j}) \\ & - (N-1)g\sum_{j=1}^{N-1}\frac{1}{j} - \sum_{i=1}^{N-1} c_i \leq c_N - (\frac{1}{N} - g) \end{aligned}$$

Hence, $M < \frac{1}{N} - c_1 - g + (1+g)\sum_{j=1}^{N-1}\frac{1}{j} \leq (1+g)\ln N + C'$ ■

VI. Fairness, Delay and Delay Jitter Measures

In the previous sections, we have obtained the bounds on the accumulated credit for MCF and FMCF. In this section, we will analyze the fairness, delay and delay jitter properties by considering their relationship to the range of accumulates credits. As will be seen, MCF and FMCF can provide $O(\log N)$ fairness, delay and delay jitter guarantees.

A. Fairness

There are two commonly used fairness measures in the literature: Golestani measure [7] and Bennet-Zhang measure [4]. While the former compares the relative amount of service received by two different flows, the latter compares the absolute amount of service a flow would receive in the ideal model and the service it receives in the designed algorithm. In this paper we adopt the more accurate Bennet-Zhang measure for analyzing the fairness property of MCF and FMCF.

We have the following theorem regarding the fairness properties of the two algorithms.

Theorem 7 (Fairness guarantee): During any time interval, the difference between the service a flow would receive in MCF or FMCF and that in the ideal fairness model is bounded by

$$\frac{1}{N} - 1 \leq \textit{service difference in MCF} < \ln N + C \qquad \text{or,}$$

$$\frac{1}{N} - g - 1 \leq \textit{service difference in FMCF} < (1+g)\ln N + C'$$

Proof. By the definitions of credit and balance, credit $c_i(t)$ is the bandwidth flow f_i would receive at slot t in the ideal model, and balance $b_i(t)$ is the bandwidth f_i actually uses at slot t in MCF or FMCF. Therefore, the accumulated credit is exactly the service difference, and the bounds of the available credit are also the bounds of the service difference. ■

B. Delay

Besides fairness, packet delay is another important performance measure for a practical fair scheduling algorithm. The *delay* of a packet is the time interval from the slot when a packet enters the queue of its flow to the slot it is sent through the gateway. The following theorem gives the packet delay of MCF and FMCF.

Theorem 8 (Delay guarantee): The packet delay of flow f_i in MCF or FMCF is bounded by

$$\begin{aligned} & \max\left\{0, \frac{\frac{1}{N} - 1 - \ln N - C + k}{c_i}\right\} \\ & < \textit{packet delay in MCF} < \frac{\ln N + C - \frac{1}{N} + 1 + k}{c_i} \qquad \text{or,} \\ & \max\left\{0, \frac{\frac{1}{N} - g - 1 - (1+g)\ln N - C' + k}{c_i}\right\} \\ & < \textit{packet delay in FMCF} \\ & < \frac{(1+g)\ln N + C' - \frac{1}{N} + g + 1 + k}{c_i} \end{aligned}$$

where k is the position of the packet when it enters the queue of f_i.

Proof. The proof for FMCF is similar to that for MCF, and we use LB and UB to represent the lower bound and upper bound respectively, of the available credit in MCF or FMCF.

Suppose that a packet of flow f_i arrives at slot t_1 is placed at the k^{th} position of the queue, and leaves the gateway at slot t_2. Then, by (3) of Property 2,

$$A_i(t_2+1) - A_i(t_1+1) = C - B$$

where C is the sum of the credit in the inclusive interval $[t_1+1, t_2]$, $C = (t_2 - t_1) \times c_i$, and B is the sum of the balances in the same interval, $B = k$. Thus,

$$(t_2 - t_1) \times c_i - k = A_i(t_2+1) - A_i(t_1+1)$$

$$\frac{LB - UB + k}{c_i} < t_2 - t_1 < \frac{UB - LB + k}{c_i}$$

Since the packet delay is always greater than 0, the low bound should be adjusted to $\max\left\{0, \frac{LB-UB+k}{c_i}\right\}$. Replacing LB and UB with the actual bounds of MCF or FMCF, we obtain the above results. ■

C. Delay Jitter

In the ideal fairness model, after a packet enters the queue of the flow, its departure time can be accurately predicated because each flow has a logically separate and independent transmission channel. When a practical scheduler is used, the scheduling sequence cannot be predetermined, and therefore the time from the packet enters the queue until it leaves the gateway may vary from packet to packet. *Delay jitter* describes the fluctuation of transmission time of packets, and is defined as the difference between the delay of a packet in MCF or FMCF and the delay in the ideal fairness model.

Theorem 9 (Delay jitter guarantee): The delay jitter of any packet of flow f_i is in MCF or FMCF is bounded by

$$\frac{\frac{1}{N} - 1 - \ln N - C}{c_i} < \textit{delay jitter in MCF} < \frac{\ln N + C - \frac{1}{N} + 1}{c_i} \quad \text{or,}$$

$$\frac{\frac{1}{N} - g - 1 - (1+g)\ln N - C'}{c_i} < \textit{delay jitter in FMCF} < \frac{(1+g)\ln N + C' - \frac{1}{N} + g + 1}{c_i}$$

Proof. The proof for FMCF is similar to that for MCF, and again we use LB and UB to represent the lower bound and upper bound respectively, of the available credit in MCF or FMCF.

Suppose a packet of flow f_i arrived at slot t_1 is placed at k^{th} position of the queue, and leaves the gateway at slot t_2. Then, in the ideal model, $t_2 - t_1 = \frac{k}{c_i}$. And by the definition,

$$\textit{delay jitter} = \textit{packet delay} - \frac{k}{c_i}$$

Applying Theorem 8,

$$\frac{LB - UB}{c_i} < \textit{delay jitter} < \frac{UB - LB}{c_i}$$

Replacing LB and UB with the actual bounds of MCF or FMCF, we obtain the above results. ■

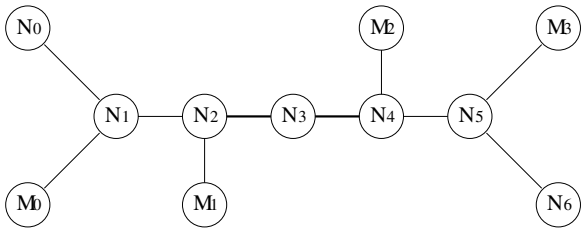


Fig. 4. The network configuration of the simulation experiments.

VII. Simulation Results

We implemented MCF and FMCF in NS2 simulator [21] to compare the end to end delay of the algorithms with other fair scheduling algorithms. For FMCF, we have several different implementations with different granularity value g to see its effect to the performance of the algorithm. Two existing algorithms, WFQ [3] and DRR [9], are compared.

As mentioned in Section II, WFQ belongs to the category of time stamp based fair scheduling algorithms. Although WFQ has good performance in fairness and delay guarantee, its time complexity is $O(\log N)$, and therefore is impractical for a large number of flows. As will be seen, compared to WFQ, our simulation results demonstrate that MCF has better end to end delay with the same complexity, and FMCF achieves comparable performance with constant complexity.

DRR is a round robin based fair scheduling algorithm, as implied by its name. Because of the round robin nature, the time complexity of DRR is $O(1)$, but its delay guarantee is $O(N)$. Compared to DRR, we demonstrate that FMCF has much better performance even with the same complexity.

The network configuration of our simulation experiment is shown in Fig. 4. There are eleven nodes, N_0 to N_6 and M_0 to M_3. The duplex links N_0N_1, N_1N_2, N_4N_5, N_5N_6, M_0N_1, M_1N_2, M_2N_4, and M_3N_5 have bandwidth of 700K bps and delay of 1ms, and the duplex links N_2N_3 and N_3N_4 have bandwidth of 1M bps and delay of 2ms. We tested the end to end delay of eleven CBR flows from N_0 to N_6 with reserved bandwidth from 5K bps to 105K bps in an increment step of 10. The eleven CBR flows behave properly and generate data in the rates equal to their reserved bandwidths. In order to congest the network, we also added three other CBR flows from M_0 to M_3, from M_0 to M_2, and from M_3 to M_1. All the three ill-behaved flows have traffic rate of 100K bps, but are only assigned reserved bandwidth of 10K bps. The excessive bandwidth of a link is allocated to each flow proportional to its reserved bandwidth.

The comparison of the average end to end delay is shown in Fig. 5(a). Since the curves of MCF and FMCF in Fig. 5(a) are very close, we give a more detailed view in Fig. 5(b) to show the differences. It can be seen that MCF outperforms both WFQ and DRR. The performance of FMCF with $g = 0.1$ is almost identical to that of MCF, while it achieves $O(1)$ time complexity. When the approximation granularity g of FMCF becomes larger, the average end to end delay of the flows with small reserved bandwidth decreases while that of the flows

TABLE 5

VARIANT OF MCF FOR SCHEDULING VARIABLE LENGTH PACKETS

```
for each flow f_i do {
    initialize accumulated credit A_i = 0;
}

while true do {
    find a flow, say, f_k, with the largest accumulated credit,
        A_k ≥ A_i for 1 ≤ i ≤ N;
    flow f_k sends a packet with length l through the gateway;

    for each flow f_i do {
        update accumulated credit by A_i = A_i + c_i × l;
    }

    flow f_k updates its accumulated credit by A_k = A_k − l;
}
```

with large reserved bandwidth increases. The reason is that the flows with small reserved bandwidth have better chances to be scheduled because of the approximation. Even when $g = 1$, which means that FMCF needs only $\lceil \frac{2+g}{g} \rceil = 3$ holes and is cheaply implementable, its performance is comparable to WFQ and much better than DRR. However, in WFQ, as the reserved bandwidth decreases, the average end to end delay increases dramatically. The reason is that WFQ schedules packets in the order of their leaving time in the ideal model, and if two packets with the same size arrive at the same time, the packet of the flow with smaller reserved bandwidth leaves the gateway later. We observe that the flows with different reserved bandwidths under DRR have roughly the same average delay. This is because that the scheduling runs in a round robin mode, and each flow has to wait for a full cycle before sending packets.

The maximum end to end delay comparison is shown in Fig. 5(c). The result is similar to that of the average delay. MCF outperforms WFQ and DRR, and FMCF closely matches MCF with lower time complexity. WFQ favors large bandwidth flows, while DRR is not sensitive to the reserved bandwidth of a flow.

VIII. SOME DISCUSSIONS

In this section, we discuss some issues related to the generalizations of MCF and FMCF and the excessive bandwidth processing in MCF and FMCF.

A. Generalization to Variable Length Packet Scheduling

Until now, we have only discussed the scheduling of fixed length packets. However, MCF (or FMCF) can also be easily adapted to apply to variable length packets, as described in Table 5. The only modifications are to use the accumulated credit instead of the available credit as the selection criterion, and to scale the effective credit and balance by a factor of the length of the transmitted packet.

B. Dealing with Excessive Bandwidth

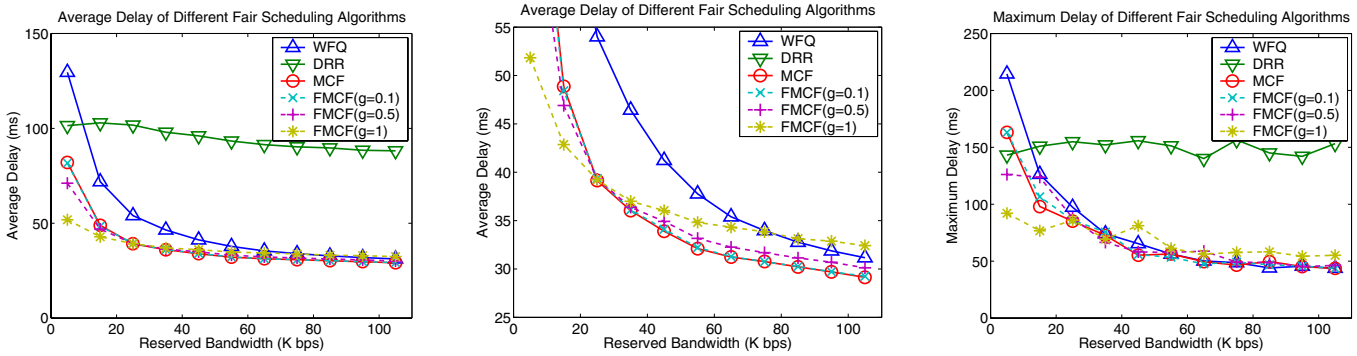
When a flow temporarily does not have packets to send, its reserved bandwidth cannot be utilized. We call the unused bandwidth of an idle flow the excessive bandwidth. In our earlier analysis, we assume that the available bandwidth of the gateway is fully utilized. Therefore, when the sum of the credit of a flow and its accumulated credit is larger than the queue length, i.e., $A_i(t) + c_i(t) >$ *queue length*, we need to adjust the credit $c_i'(t) =$ *queue length* $- A_i(t)$, so as to let the flow have the exact amount of available credit to transmit its packets. And the excessive bandwidth should be reallocated to fully utilize the potential transmission capacity.

In fact, MCF or FMCF (with a slight modification of putting a flow f_i with $V_i(t) > lastV - 1 + g \times \lceil \frac{2+g}{g} \rceil$ into $hole[\lceil \frac{2+g}{g} \rceil]$) can work well without reallocating the excessive bandwidth, since they have the property of self-converge. It is possible that an idle flow temporarily may accumulate some credit, and a flow with negative available credit may continue to consume the bandwidth and decrease the accumulated credit further. However, as soon as the idle flow becomes backlogged again, it will consistently gets scheduled and its accumulated credit will decrease accordingly due to its large accumulated credit. On the contrary, those flows with negative accumulated credits will not be allowed to transmit, so as to recover their accumulated credits. This way, the accumulated credits of both types of flows are heading towards zero.

Even the excessive bandwidth is to be reallocated, it can be done in $O(1)$ time. For example, a simple way is to assign credit only to the guaranteed performance flows ($f_i, i \neq 1$) corresponding to their requested bandwidth, and not to give the best effort service flow f_1 any credit. Accordingly, the guaranteed performance flows are given higher priority and are scheduled as long as they have positive available credits. Only when all the guaranteed performance flows have negative available credits, the best effort flow can be scheduled. Therefore, all the excessive bandwidth is automatically assigned to the best effort flow, while other flows also receive guaranteed services. And this approach requires only constant time complexity.

IX. CONCLUSIONS

In this paper, we have proposed a new fair scheduling algorithm, Most Credit First (MCF). MCF emulates the ideal fairness model by tracking the bandwidth usage difference between that a flow would receive in the ideal model and that it actually receives in MCF. MCF exhibits nice $O(1)$ fairness and delay guarantees in simulations, and has theoretical $O(\log N)$ performance guarantees. In addition, to further lower the time complexity of MCF, we proposed an approximation algorithm of MCF, Fast Most Credit First (FMCF). By utilizing approximation and synchronization techniques, FMCF achieves $O(1)$ time complexity, and at the same time preserves the theoretical $O(\log N)$ performance guarantees of MCF. We also analyzed the fairness, delay and delay jitter properties of MCF and FMCF by considering their relationship to the range of the accumulated credit. We implemented MCF and FMCF in NS2 simulator to compare their end to end delay with other existing fair scheduling algorithms. Our experimental results demonstrate that MCF outperforms WFQ and DRR in the end to end delay performance, and FMCF successfully matches the performance of MCF with constant time complexity.



(a) Average delay (complete figure) (b) Average delay (detailed view) (c) Maximum delay

Fig. 5. The average delay and maximum delay of CBR flows in different fair scheduling algorithms

ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0073085, CCR-0207999 and ECS-0427345.

REFERENCES

[1] M. Garett, "A service architecture for ATM: from applications to scheduling," *IEEE Network Magazine*, pp. 6-14, May 1996.

[2] 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.

[3] 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.

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

[5] J. Bennet, H. Zhang, "Hierarchical packet fair queueing algorithms," *ACM SIGCOMM '96*, pp. 143–156, Palo Alto, CA, Aug. 1996.

[6] P. Goyal, H. Vin, H. Cheng, "Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Networking*, vol. 5, no. 5, pp. 690-704, Oct. 1997.

[7] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," *IEEE INFOCOM '94*, pp. 636-646, Toronto, Canada, Jun. 1994.

[8] L. Zhang, "Virtual clock: a new traffic control algorithm for packet switching networks," *ACM SIGCOMM '90*, pp. 19-29, Philadelphia, PA, Sep. 1990.

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

[10] 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.

[11] N. Matsufuru, R. Aibara, "Efficient fair queueing for ATM networks using uniform round robin," *IEEE INFOCOM '99*, pp. 389-397, New York, Mar. 1999.

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

[13] 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.

[14] P. Goyal, H. Vin, "Generalized guaranteed rate scheduling algorithms: a framework," *IEEE/ACM Trans. Networking*, vol. 5, no. 4, pp. 561-571, Aug. 1997.

[15] D. Stiliadis, A. Varma, "Efficient fair queueing algorithms for packet switched networks," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375-385, Jun. 1998.

[16] D. Stiliadis, A. Varma, "Rate proportional servers: a design methodology for fair queueing algorithms," *IEEE/ACM Trans. Networking*, vol. 6, no. 2, pp. 164-174, Apr. 1998.

[17] G. Xie, S. Lam, "Delay guarantee of virtual clock server," *IEEE/ACM Trans. Networking*, vol. 3, no. 6, pp. 683-689, Dec. 1995.

[18] N. Figueira, J. Pasquale, "An upper bound on the delay for the virtual clock service discipline," *IEEE/ACM Trans. Networking*, vol. 3, no. 4, pp. 399-408, Aug. 1995

[19] N. Figueira, J. Pasquale, "A schedulability condition for deadline-based service disciplines," *IEEE/ACM Trans. Networking*, vol. 5, no. 2, pp. 232-244, Apr. 1997.

[20] J. Conway, R. Guy, *The Book of Numbers*, New York: Springer-Verlag, pp. 143 and 258-259, 1996.

[21] http://www.isi.edu/nsnam/ns/