

## A Constant Complexity Fair Scheduler with $O(\log N)$ Delay Guarantee

Deng Pan<sup>1</sup> and Yuanyuan Yang<sup>2</sup>

<sup>1</sup> Department of Computer Science  
State University of New York at Stony Brook, Stony Brook, NY 11790  
dengpan@cs.sunysb.edu

<sup>2</sup> Department of Electrical and Computer Engineering  
State University of New York at Stony Brook, Stony Brook, NY 11790  
yang@ece.sunysb.edu

**Abstract.** Many Internet multimedia applications require the support of network services with fairness and delay guarantees. Currently, there are two types of fair schedulers in the literature. The time stamp based schedulers achieve good fairness and delay guarantees but have high  $O(\log N)$  time complexity, where  $N$  is the number of incoming flows. While the round robin based schedulers reach  $O(1)$  complexity, their delay guarantees are  $O(N)$ . Aiming at constant time complexity as well as good fairness and delay guarantees, we design a new fair scheduler suitable for variable length packets in this paper. Fast Credit Based (FCB) fair scheduling, the algorithm we propose, provides  $O(\log N)$  fairness and delay guarantees, by tracking and minimizing the difference between the service a flow reserves and that it actually receives. It reduces the time complexity to  $O(1)$  by utilizing approximation and synchronization. To compare FCB with other fair schedulers on their end-to-end delay performance, simulations are conducted in NS2 for various packet lengths, and the results show that FCB achieves short end-to-end delay and handles variable length packets efficiently.

**Keywords:** Fair scheduling, shared output link, bandwidth guarantee, delay guarantee.

### 1 Introduction

The booming Internet multimedia applications require the next generation of networks to provide services with bandwidth and delay guarantees, besides the traditional best effort services. A fair scheduler works at a shared output link or gateway to provide incoming flows with guaranteed bandwidth and delay performance. It ensures that the difference between the services that any flow reserves and that it actually receives is bounded within a specified range, regardless of the length of the time interval.

Fair schedulers in the literature can be classified into three types: 1) *Time stamp based*. Time stamp based fair schedulers, such as *WFQ* [1] and *WF<sup>2</sup>Q* [2], compute time stamps for each incoming packet, and schedule packets in the order of the time stamps. They usually provide excellent fairness and delay guarantees, but have high  $O(\log N)$  time complexity due to the operation to sort packets, where  $N$  is the number of flows of the gateway. 2) *Round robin based*. Round robin based fair schedulers, such as *DRR* [3] and *SRR* [4], serve the flows in a round robin manner, so that each flow has an equal opportunity of consuming bandwidth. They

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. 3) *Combination of both*. Some recently proposed algorithms, such as *BSFQ* [5] and *Stratified Round Robin* [6], attempt to obtain the tight delay bound of time stamp schedulers as well as the low complexity of round robin schedulers, by adopting a basic round robin like scheduling policy plus time stamp based scheduling on a reduced number of units. They improve the time complexity by reducing the number of items that need to be sorted, but still have  $O(N)$  worst case delay because of the round robin nature.

The following factors must be considered to design a good fair scheduler. 1) *Bandwidth guarantee*. The scheduler should ensure each flow its reserved bandwidth, so that well-behaved flows can be protected from malicious behavior. 2) *Delay guarantee*. Bandwidth guarantee should be obtained in an efficient way, and therefore well-behaved flows can have short and bounded packet delay. 3) *Low complexity*. The scheduler should have low time complexity to achieve high speed scheduling. Especially, constant time complexity enables the scheduler to maintain performance when the number of flows increases. 4) *Capability to schedule variable length packets*. Some schedulers, such as *SRR* [4], consider only fixed length packets, and can not handle variable length packets efficiently. They have to segment the incoming variable length packets into fixed length cells before scheduling, and the receivers also need extra buffer space to reassemble the segmentations back to the original packets.

In this paper we aim at designing a fair scheduler for variable length packets with constant time complexity as well as good fairness and delay guarantees. *Fast Credit Based (FCB)* fair scheduling, the algorithm we propose, adopts a credit based policy, and provides fairness and delay guarantees by tracking and minimizing the difference between the service a flow should receive and that it has actually received in the algorithm. FCB achieves  $O(1)$  complexity by utilizing approximation and synchronization, and we theoretically prove that FCB provides  $O(\log N)$  fairness and delay guarantees. To compare FCB with other fair schedulers on their end-to-end delay performance, simulations are conducted in NS2 with packet length following different statistical distributions, and the results show that FCB matches WFQ with reduced complexity, and that FCB handles variable length packets efficiently.

## 2 Fast Credit Based Fair Scheduling

In this section, we present the *Fast Credit Based (FCB)* fair scheduling algorithm. We first introduce some definitions, and then describe the algorithm. In the following, a gateway (or shared output link) with  $N$  incoming flows  $\{f_1, \dots, f_N\}$  is considered.

### 2.1 Terminologies

The *reservation*  $r_i$  of flow  $f_i$  is its reserved bandwidth normalized with respect to the total bandwidth of the gateway. By the definition, we have that  $0 < r_i \leq 1$ , and to avoid overbooking,  $\sum_{i=1}^N r_i \leq 1$ .

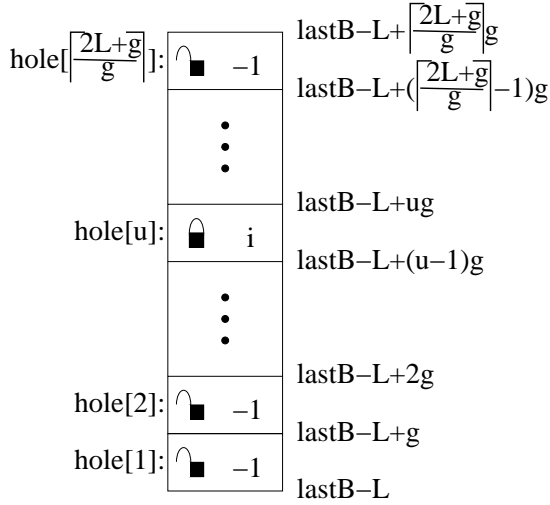
The *credit*  $c_i(t)$  of flow  $f_i$  is the rate that the flow should ideally receive bandwidth from the gateway according to its reservation at time  $t$ , i.e.,

$$c_i(t) = \begin{cases} \frac{r_i(t)}{\sum_{j \in \Delta(t)} r_j(t)}, & \text{flow } f_i \text{ is backlogged at time } t \\ 0, & \text{otherwise} \end{cases}$$

where  $\Delta(t)$  is the set of backlogged flows at time  $t$ .

There may be the case that the sum of the reservations of all the backlogged flows is less than the total bandwidth of the gateway. As in the ideal GPS [7] fairness system, the excessive bandwidth is reallocated to avoid wasting available transmission capacity. Therefore, if there is





**Fig. 1.** FCB uses synchronization holes to achieve flow approximation. The flows whose balances are in the same range compete for a single hole.

```

for each flow  $f_i$  do {
  initialize balance  $b_i = 0$ ;
}
 $lastB = 0$ ;
while true {
  for ( $j = 1; j \leq \lceil \frac{2L+g}{g} \rceil; ++j$ ) {
     $hole[j] = -1$ ;
  }
  for each flow  $f_i$  do {
    if ( $hole[\lceil \frac{b_i - lastB + L}{g} \rceil] == -1$ )
       $hole[\lceil \frac{b_i - lastB + L}{g} \rceil] = i$ ;
  }
   $k = -1$ ;
  for ( $j = 1; j \leq \lceil \frac{2L+g}{g} \rceil; ++j$ ) {
    if ( $hole[j] \neq -1$ )  $k = hole[j]$ ;
  }
  flow  $f_k$  sends a packet through the gateway,
  say, from  $t_a$  to  $t_{a+1}$ ;
   $lastB = b_k$ ;
  for each flow  $f_i$  do {
    update balance by
       $b_i + = \int_{t_a}^{t_{a+1}} c_i(x)dx - \int_{t_a}^{t_{a+1}} d_i(x)dx$ ;
  }
}

```

**Fig. 2.** Pseudo code description of the FCB fair scheduling algorithm.

filled holes and granted to transmit a packet, and its balance value is assigned to  $lastB$  for the next round of scheduling. After the packet transmission, the balance of each flow is updated accordingly. Although the balance update formula includes integral computation, which seems not efficient to implement, the credit and debit values are actually fixed during each scheduling point interval  $([t_a, t_{a+1}))$ , and therefore the integral can be simply computed as multiplication.

**Theorem 1** *If  $lastB$  is at most  $g$  less than the largest balance of all the flows at time  $t_{a-1}$ , then the largest balance at time  $t_a$  is in the range  $(lastB - L, lastB + L + g)$ .*

**Proof.** Suppose that at time  $t_{a-1}$ , flow  $f_j$  has the largest balance,  $b_j(t_{a-1}) \geq b_i(t_{a-1})$  for any  $1 \leq i \leq N$ , and  $f_k$  is selected to transmit. Then,  $lastB = b_k(t_{a-1})$  for time  $t_a$ , and because  $b_k(t_{a-1})$  is at most  $g$  less than  $b_j(t_{a-1})$ , we obtain  $lastB + g \geq b_j(t_{a-1})$ .

Also suppose that at time  $t_a$ , flow  $f_l$  has the largest balance,  $b_l(t_a) \geq b_i(t_a)$  for any  $1 \leq i \leq N$ . We prove that  $lastB - L < b_l(t_a) \leq lastB + L + g$ .

On the one hand, when  $t \in [t_{a-1}, t_a)$ ,  $d_j(t) \leq 1$ , and we have

$$b_l(t_a) \geq b_j(t_a) = b_j(t_{a-1}) + \int_{t_{a-1}}^{t_a} c_j(x)dx - \int_{t_{a-1}}^{t_a} d_j(x)dx > b_j(t_{a-1}) - \int_{t_{a-1}}^{t_a} d_j(x)dx \geq b_j(t_{a-1}) - L$$

And by  $b_j(t_{a-1}) \geq b_k(t_{a-1})$ , it follows that  $b_l(t_a) > b_k(t_{a-1}) - L = lastB - L$ .

On the other hand, when  $t \in [t_{a-1}, t_a)$ ,  $d_l(t) \geq 0$ , and we can obtain

$$b_l(t_a) = b_l(t_{a-1}) + \int_{t_{a-1}}^{t_a} c_l(x)dx - \int_{t_{a-1}}^{t_a} d_l(x)dx \leq b_l(t_{a-1}) + \int_{t_{a-1}}^{t_a} c_l(x)dx \leq b_l(t_{a-1}) + L$$

Using the fact that  $b_l(t_{a-1}) \leq b_j(t_{a-1})$  and  $b_j(t_{a-1}) \leq lastB + g$ , we obtain  $b_l(t_a) \leq lastB + L + g$ .

Next, we show that the precondition of the theorem, that the balance of the scheduled flow is at most  $g$  less than the largest balance, is always guaranteed by FCB. Since the balance value of all the holes is in the range  $(lastB - L, lastB - L + \lceil \frac{2L+g}{g} \rceil g)$ , and  $lastB - L + \lceil \frac{2L+g}{g} \rceil g \geq lastB + L + g$ , from the above proof, we know that the balance  $b_l(t_a)$  of  $f_l$  must be in the



Suppose that flow  $f_k$  transmits a packet during the interval  $[t_{a-1}, t_a)$ . According to the scheduling policy,  $b_k(t_{a-1})$  is at most  $g$  less than the largest balance at time  $t_{a-1}$ , and from Theorem 2, the largest balance at any time should be greater than or equal to zero. Therefore, we have  $b_k(t_{a-1}) \geq -g$ . And  $b_k(t_a) = b_k(t_{a-1}) + \int_{t_{a-1}}^{t_a} c_k(x)dx - \int_{t_{a-1}}^{t_a} d_k(x)dx$ . It is obvious that  $\int_{t_{a-1}}^{t_a} c_k(x)dx \geq 0$  and  $\int_{t_{a-1}}^{t_a} d_k(x)dx \leq L$ , and we obtain  $b_k(t_a) \geq -g - L$ . ■

Next, we derive the upper bound of the balance. 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  $c_i(t) = c_i$ . Because the excessive bandwidth of the idle flows is reallocated to other backlogged flows which we are considering, this assumption does not weaken the generality of the results, but it makes the analysis much easier.

Similar to the fact that the balance of a flow decreases as it is transmitting a packet, its balance increases when it is idle, because it has positive credit and zero balance. Thus, in order to compute the upper bound of the balance, we only need to consider the time when a flow begins sending a packet.

The basic idea of the proof is to assume that the maximum value of the balance is reached by one flow at a specific time point, and consider the sum of the balances of the flows that have been recently scheduled before that specific time point. 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 3 can be applied to derive the bound of the maximum value we assumed.

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

$M$ : the maximum value of the balance.

$t_m$ : the time that the maximum balance is reached.

$F(t_a, t_m)$ : the set of flows that begin to transmit packets at scheduling points  $t_a, \dots, t_m$ , where  $t_a \leq t_m$ . Without loss of generality, we assume that flow  $f_1$  reaches the maximum balance  $M$  at time  $t_m$ . Then,  $f_1$  must begin to transmit at  $t_m$ , otherwise  $b_1(t_{m+1}) = b_1(t_m) + \int_{t_m}^{t_{m+1}} c_1 dx - \int_{t_m}^{t_{m+1}} d_1(x)dx = M + \int_{t_m}^{t_{m+1}} c_1 dx > M$ . Also assume that the sequence of the flows added into  $F(t_a, t_m)$  when considering the sum of the balances is  $f_1, f_2, \dots, f_N$ . In other words, when we look back from the scheduling point  $t_m$ ,  $f_2$  is the most recently scheduled flow. For example, if  $f_2$  was most recently scheduled at  $t_b(t_b < t_m)$  before time  $t_m$ , we have  $F(t_m, t_m) = \{f_1\}$ ,  $F(t_{b+1}, t_m) = \{f_1\}$ , and  $F(t_b, t_m) = \{f_1, f_2\}$ . If  $f_3$  was most recently scheduled at  $t_c(t_c < t_b)$ , then  $F(t_{c+1}, t_m) = \{f_1, f_2\}$  and  $F(t_c, t_m) = \{f_1, f_2, f_3\}$ .

$T(t_a, t_m)$ : the sum of the balances of all the flows in  $F(t_a, t_m)$  at time  $t_a$ , i.e.,  $T(t_a, t_m) = \sum_{f_i \in F(t_a, t_m)} b_i(t_a)$ .

We next introduce some supporting lemmas. Due to space limitation, the proofs of these lemmas are omitted.

**Lemma 2** Suppose  $F(t_a, t_m) = \{f_1, \dots, f_k\}$  and  $F(t_{a-1}, t_m) = \{f_1, \dots, f_k, f_{k+1}\}$ . Then,

$$T(t_{a-1}, t_m) \geq \frac{k+1}{k} \left( T(t_a, t_m) - L \sum_{i=1}^k c_i \right) - g$$

**Lemma 3** Suppose  $F(t_a, t_m) = \{f_1, \dots, f_k\}$  and  $F(t_{a-1}, t_m) = F(t_a, t_m) = \{f_1, \dots, f_k\}$ . Then,

$$T(t_{a-1}, t_m) \geq T(t_a, t_m)$$

**Lemma 4** Suppose  $F(t_m, t_m) = \{f_1\}$  and  $F(t_a, t_m) = \{f_1, \dots, f_k\}$ . Then,

$$T(t_a, t_m) \geq kM - kL \sum_{i=1}^{k-1} \left( c_i \sum_{j=i}^{k-1} \frac{1}{j} \right) - kg \sum_{i=1}^{k-1} \frac{1}{i+1}$$

The upper bound on the balance is given in the following theorem.



### 3.3 Delay Guarantee

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

**Theorem 6** *The packet delay of flow  $f_i$  in FCB is bounded by*

$$\max \left\{ 0, \frac{-g - L - (L + g) \ln N - C + q}{c_i} \right\} < \text{packet delay} < \frac{(L + g) \ln N + C + g + L + q}{c_i}$$

where  $q$  is the queue length of  $f_i$  after the packet arrives.

**Proof.** Suppose that a packet of flow  $f_i$  arrives at time  $t_a$  and leaves the gateway at time  $t_b$ , and that the queue length is  $q$  after this packet is put in the queue. Then, by the definition of the balance, we have

$$b_i(t_b) - b_i(t_a) = \int_{t_a}^{t_b} c_i dx - \int_{t_a}^{t_b} d_i(x) dx = c_i(t_b - t_a) - q$$

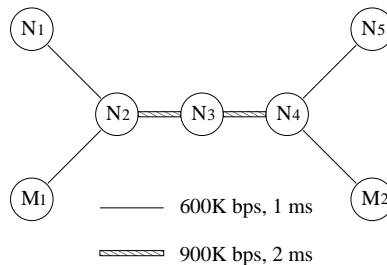
Because  $-g - L - (L + g) \ln N - C < b_i(t_b) - b_i(t_a) < (L + g) \ln N + C + g + L$ , we obtain

$$\frac{-g - L - (L + g) \ln N - C + q}{c_i} < t_b - t_a < \frac{(L + g) \ln N + C + g + L + q}{c_i}$$

Since the packet delay is always greater than 0, the lower bound should be adjusted if it is smaller than 0. ■

## 4 Simulation Results

We compare FCB with other fair schedulers on their end-to-end delay performance by simulation in NS2 [10]. Two implementations of FCB with  $g = L/10$  and  $g = L/3$  respectively are included to see the effect of the granularity value on the algorithm. Also, since FCB is designed to handle variable length packets, we conducted the simulations under the traffic with packet length following different statistical distributions: constant packet length (Figure 4(a)), packet length following uniform distribution (Figure 4(b)), and packet length following normal distribution (Figure 4(c)). The maximum packet length  $L$  is set to 300 bytes in all the situations. WFQ



**Fig. 3.** The network topology in the simulations.

and DRR are used as the comparison counterparts. WFQ is the earliest and a typical time stamp based fair scheduler. Although WFQ has good fairness and delay guarantees, it has a high  $O(\log N)$  time complexity. By comparing with WFQ, we show that FCB matches the performance of WFQ with reduced complexity. DRR is a representative of the round robin based scheduler family. It has  $O(1)$  time complexity but  $O(N)$  worst case delay. By comparing with DRR, we demonstrate that, with the same complexity, FCB achieves better performance than DRR.

The network topology for the simulations is illustrated in Figure 3. We set up ten CBR flows between  $N_1$  and  $N_5$  with reserved bandwidth from 10K bps to 100K bps in an increment





tween the service the flow should receive and actually receives. By introducing approximation and synchronization, FCB successfully reduces the time complexity to  $O(1)$ . Also, we theoretically prove that FCB provides  $O(\log N)$  fairness and delay guarantees. Finally, simulations are conducted to compare the end-to-end delay performance of FCB with those of WFQ and DRR, and the results show that FCB matches the performance of WFQ with reduced complexity, and that FCB is able to efficiently schedule variable length packets.

### Acknowledgement

The research work was supported in part by the U.S. National Science Foundation under grant numbers CCR-0073085 and CCR-0207999.

### References

1. 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.
2. H. Zhang, "WF2Q: worst-case fair weighted fair queueing," *IEEE INFOCOM '96*, pp. 120-128, San Francisco, CA, Mar. 1996.
3. M. Shreedhar, G. Varghese, "Efficient fair queueing using deficit round robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375-385, Jun. 1996.
4. 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.
5. S. Cheung and C. Pencea, "BSFQ: bin sort fair queueing," *IEEE INFOCOM '02*, pp. 1640-1649, New York, Jun. 2002.
6. 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.
7. 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.
8. J. Conway, R. Guy, *The Book of Numbers*, New York: Springer-Verlag, pp. 143 and 258-259, 1996.
9. S. Golestani, "A self-clocked fair queueing scheme for broadband applications," *IEEE INFOCOM '94*, pp. 636-646, Toronto, Canada, Jun. 1994.
10. <http://www.isi.edu/nsnam/ns/>