Proceedings of the Fourth IASTED International Conference
Communications, Internet, and Information Technology
October 31 - November 2, 2005, Cambridge, USA

SPEEDING UP THE CONVERGENCE OF ESTIMATED FAIR SHARE IN CSFQ

Peng Wang
Department of Electrical & Computer Engineering
University of Delaware
Newark, DE, USA
email: pwangee@udel.edu

David L. Mills
Department of Electrical & Computer Engineering
University of Delaware
Newark, DE, USA
email: mills@ece.udel.edu

ABSTRACT
Core-stateless Fair Queueing (CSFQ) is a scheme to achieve approximate fair bandwidth sharing without per-flow state in the interior routers. The extra packets that beyond the fair share for each flow are dropped probabilistically based on the attached flow rate in the packet header and the estimated fair share. A heuristic method is used to estimate the fair share in CSFQ. In our previous work CSFQIMP (CSFQ Improvement), we took the probabilistic idea from SRED (Stabilized RED) and applied it in CSFQ to estimate the fair share. The probabilistic approach achieves a comparable or even better performance than the original heuristic approach. However, the convergence speed of the probabilistic approach is slow. Therefore, we propose a method to speed up the convergence of the fair share estimate in this paper. The idea comes from that the router randomly selects k packets instead of one packet to compare with the incoming packet in SRED. We show that the convergence speed is increased in the usual cases. Simulation results show that the speedup approach achieves a quick convergence compared with the original probabilistic method in our previous work.

KEY WORDS
CSFQ, Fairness, Convergence, Network Management

1 Introduction

The current Internet provides a connectionless, best-effort, and end-to-end packet service by using the IP protocol. The majority of the Internet traffic, including HTTP, FTP, TELNET, and email traffic, is carried by TCP protocol. With the increasing greediness of unresponsive multimedia applications over the Internet, one must be concerned with the fairness problem.

The fair bandwidth allocations can isolate flows and protect well-behaved flows from ill-behaved ones. The definition of flows is very flexible. In this paper, the flow is defined by 5-tuple (source IP address, destination IP address, source port, destination port, and protocol) in the packet header. In CSFQ [1] edge routers maintain the per-flow state, and core routers are stateless. Edge routers measure the per-flow rate and attach the flow rate as a label to each packet header. Core routers measure the aggregate flow rate and estimate the fair share. Then the incoming packets are dropped probabilistically based on the packet label (flow rate) and the estimated fair share. Therefore, the accuracy of the estimated fair share is the key factor in determining the performance of CSFQ. CSFQ uses a heuristic method to estimate the fair share.

In order to stabilize its buffer occupation, SRED [2] determines the dropping probability for each incoming packet based on the estimated number of flows. A Zombie list consists of a shift register that is much larger than the router's queue, and stores the recently seen packets. SRED [2] uses a probabilistic method based on a Zombie list to estimate the number of flows at the router.

Assume lots of flows shared the router, the estimated fair share is approximately set to the link bandwidth divided by the estimated number of flows. In our previous work [3], we took the probabilistic idea from SRED and applied it in CSFQ to estimate the fair share without the Zombie list because CSFQ has an important characteristic that each packet has its flow rate as a label in the packet header. The probabilistic approach achieves a comparable or even better performance than the original heuristic approach. However, the convergence speed of the probabilistic approach is slow, which makes the assumption that traffics are stationary during the estimation period suspect.

Therefore, in this paper we propose a method to speed up the convergence of the fair share estimate. The idea comes from that the router randomly selects k packets instead of one packet to compare with the incoming packet in SRED. We show that the convergence speed is increased in the usual cases. Simulation results show that the speedup approach achieves a quick convergence compared with the original probabilistic method in our previous work [3].

The remainder of the paper is structured as follows. In the next section, CSFQ architecture is described in more detail. In Section 3, we describe a probabilistic approach for achieving the fair share in detail from our previous work CSFQIMP [3]. In section 4, we describe the method to speed up the convergence of estimated fair share. In Section 5, we evaluate the performance of our method in com-

[1]This research is sponsored by the NSWCDD grant N00178-04-1-9001. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements of NSWCDD or the US goverment.

parison to the original probabilistic method from [3] by using simulations. Finally, we conclude in Section 6.

2 CSFQ Architecture

2.1 Objectives

The primary objective is to achieve max-min fairness [4] among the flows in a congested router. Consider a link with capacity C serving N flows, the flow's arrival rate is $r_i(t), i = 1, ..., N$. Let $\alpha(t)$ be the fair share rate at time t and $A(t) = \sum_{i=1}^{N} r_i(t)$ be the total arrival rate. Max-min fairness is then achieved when the fair share $\alpha(t)$ is the unique solution to:

$$C = \sum_{i=1}^{N} min(r_i(t), \alpha(t)) \quad (1)$$

If $A(t) < C$ (no congestion happens), all packets pass through the router unconstrained and the fair share $\alpha(t)$ is set to $max_i(r_i(t))$. On the other hand, the flow rate $r_i(t)$ above the fair share $\alpha(t)$ is constrained to $\alpha(t)$, while the flow rate $r_i(t)$ less than the fair share is unconstrained.

2.2 CSFQ Algorithm

To facilitate our discussion, let us introduce how CSFQ achieves the above objective in three steps.

1) Measure the Flow Arrival Rate

The flow arrival rates $r_i(t)$ measured at the edge routers are attached to the packet header as a label. The exponential average is used to calculate the flow arrival rate and updated for each incoming packet. Let t_i^k and l_i^k be the arrival time and the packet length of the kth packet of flow i.

$$r_i^{new} = (1 - e^{-T_i^k/K})\frac{l_i^k}{T_i^k} + e^{-T_i^k/K} r_i^{old} \quad (2)$$

where $T_i^k = t_i^k - t_i^{k-1}$ is the packet inter-arrival time, and K is a constant.

2) Link Fair Rate Estimation

CSFQ uses a heuristic algorithm to estimate the fair share rate. Let us introduce three variables first: α, the estimated fair share rate; A, the estimated aggregate arrival rate; F, the estimated rate of the accepted traffic. The exponential average is used to calculate A and F. Detailed descriptions for A and F can be found in [1].

Assume C is the link capacity and K_c is a window size to filter out the inaccuracy due to the exponential smoothing. If $A < C$ at all times during a time interval of length K_c, the fair share α is set to the largest rate of the active flows in the last K_c time units. On the other hand, if $A > C$ at all times during a time interval of length K_c, the fair share rate α is updated by the formula $\alpha_{new} = \alpha_{old} \cdot C/F$. Generally, the fair share α is updated at the end of the interval K_c.

Moreover, to reduce the negative effects of buffer overflow, another heuristic rule is used: α is decreased by a small fixed percentage for each buffer overflow. But α cannot be decreased more than 25% consecutively to avoid overcorrection.

3) Packet Dropping and Label Rewriting

For each incoming packet, the router calculates a dropping probability based on the packet label and the fair share estimate: $Prob = max(0, 1 - \alpha/p.label)$. The dropping algorithm limits the flows to their fair share bandwidth. Finally, the packets are relabeled using the minimum of the current packet label and the router's estimated fair share α, because the packets beyond the fair share are dropped at the routers and the original packet label is not an accurate estimate of its actual flow rate.

3 Probabilistic Approach for Fair Bandwidth Allocations

3.1 Estimate the Number of Flows in SRED

A probabilistic method to estimate the number of flows is used in the algorithm SRED without maintaining the per-flow state. SRED uses a Zombie list that is a large shift register to store several thousand recently seen packets. For each incoming packet, SRED randomly selects a packet from the Zombie list and compares these two packets. If they are from the same flow, it is called a "hit". Otherwise, it is called a "miss". The router maintains a hit-frequency count.

We assume that each arrival packet belongs to one of the N flows. Let P_i be the probability that a packet belongs to flow i, and assume P_i is stationary over the time in which the estimation is done. Then the hit probability for an incoming packet is equal to:

$$P_{hit} = \sum_{i=1}^{N} P_i^2 \quad (3)$$

Thus, if the flows have the the same traffic intensity, the hit probability P_{hit} is $1/N$. The inverse of the hit probability is an exact estimate of the number of flows. Even when the flows have asymmetric traffic intensities, the inverse of the hit probability is a reasonable estimate of the number of the active flows.

After sampling n new arrival packets, there are m hits total.

$$n \cdot \frac{1}{N} \approx n \cdot P_{hit} = m \quad (4)$$

The estimated number of flows is equal to n/m. When there are tens of thousands of flows shared the router, the estimated fair share is approximately set to the link bandwidth divided by the estimated number of flows.

3.2 Estimate the Number of Flows in CSFQ

The probabilistic idea is taken from SRED and applied to estimate the number of flows in CSFQ without the Zombie list. We assume that each arrival packet belongs to one of the N flows and the probability that an arriving packet belongs to a given flow is independent of all other packets. Let r_i be the rate of flow i stored in the packet label and A be the aggregate flow rate measured at the router. Let $\hat{P}_i = r_i/A$ denote the proportion of traffic that belongs to flow i. We assume that $\hat{P}_i$ is stationary over the time in which the estimation is done. This means that we can view $\hat{P}_i$ as the probability that an incoming packet belongs to flow i.

For n new arrival packets, there are $n \cdot P_i$ packets that belong to flow $i, i = 1, \ldots, N$ and $\sum_{i=1}^{N} P_i = 1$ (P_i is the actual probability). When a packet from flow i arrives at the router, the probability P_i is approximated by $\hat{P}_i = r_i/A$. After the $n \cdot P_i$ packets that belong to flow i arrive at the router, the sum of the probability $\hat{P}_i$ for these $n \cdot P_i$ packets is equal to $(n \cdot P_i) \cdot \hat{P}_i \approx n \cdot {P_i}^2$. Furthermore, since there are N flows, the sum of the probability $\hat{P}_i$ for each flow i is equal to $n \cdot {P_i}^2$ where $i = 1, \ldots, N$.

However, it is impossible to compute $n \cdot {P_i}^2$ for each flow i since the core router does not maintain the per-flow state and the packets cannot be classified into the different flows. But if we sum probability $\hat{P}_i$ for all n packets without classifying them, we can get a sum m:

$$m = nP_1^2 + nP_2^2 + \cdots + nP_N^2 = n\sum_{i=1}^{N} P_i^2 = nP_{hit} \quad (5)$$

This is the same as the formula (4) used in SRED. The estimated number of flows is equal to n/m. Assume lots of flows share the router, the estimated fair share is approximated set to the link bandwidth divided by the estimated number of flows.

One problem encountered by this method is an inaccuracy when the traffic density is vastly different. Thus, we consider n accepted packets instead of n incoming packets at the routers. The incoming packets are classified into three classes: successfully transmitted packets, dropped packets due to dropping policy and dropped packets due to queue overflows. The accepted packets include the successfully transmitted packets and the dropped packets due to queue overflows at the routers. Not surprising, considering the accepted packets reduces the impact of heavy flows and improves the estimated accuracy. Moreover, the estimated number of flows converges to the actual number of flows after several iterations. The following example is given to show the convergence process.

Let us consider the case that two aggregate flows arrive at a core router. The bandwidth of the core router is $5,000$ packets/sec and the packet size is fixed. The first aggregate flow consists of 500 flows with the flow rate 10 packets/sec each. The second aggregate flow consists of 5 flows with the flow rate $1,000$ packets/sec each. Thus, the aggregate arrival rate A is $10,000$ packets/sec. The $\hat{P}_i$ for each flow in the first aggregate flow is $1/1000$ and the $\hat{P}_i$ for each flow in the second aggregate flow is $1/10$. We set the initial value of the estimated fair share is $5,000$ packets/sec. The estimated number of flows is updated once each second at the core router.

The estimated number of flows converges to 505 after 12 iterations. Figure 1 gives the convergence process of the estimate. This shows that the probabilistic method is reasonable and converges if the flows are stationary. However, the convergence speed is slow.

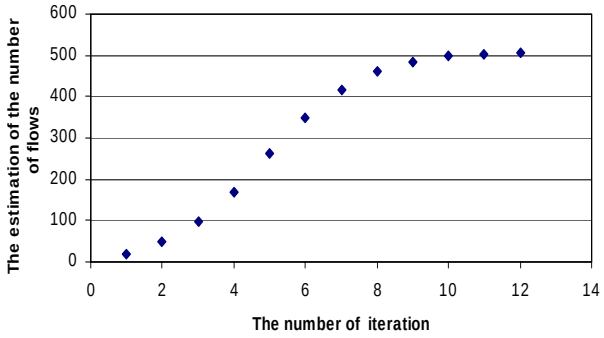

Figure 1. The convergence of the estimated number of flows

4 Speeding up the Convergence of Estimated Fair Share

In SRED, when there are tens of thousands of flows, the convergence speed of the estimated number of flows is slow due to lots of misses. In general, we can choose $k > 1$ packets from the Zombie list, and compare all of them with the incoming packet. Not surprisingly, the convergence of the estimated number of flows speeds up. Since the reverse of the hit probability is the estimated number of flows, we only study the convergence of the hit probability in the following sections.

4.1 The convergence of the hit probability for choosing one packet in SRED

The hit probability for an incoming packet is equal to $P_{hit} = \sum_{i=1}^{N} P_i^2$ for choosing one packet from the Zombie list to compare. RATE [5] provides a method to estimate the sampling size in order to achieve the required estimated accuracy. Let us determine a sample size T that the estimate $\hat{P}_{hit} \in (P_{hit} - \frac{\beta}{2}, P_{hit} + \frac{\beta}{2})$ with the probability greater than α. In other words, we are willing to tolerate an error of $\pm\frac{\beta}{2}$ with the probability less than α. Z_α is used to denote the α percentile for the unit normal distribution. For example, if $\alpha = 99.99\%$, then $Z_\alpha = 4.0$.

Theorem 1: Let $Y(T)$ represents the number of hits

after T samples. For large T,

$$\sqrt{T}\,[\frac{Y(T)}{T} - P_{hit}] \sim N[0, P_{hit}(1-P_{hit})] \quad (6)$$

where $N[a,b]$ represents a normal distribution with mean a and variance b.

Proof: Let $y_j, j = 1,\ldots,T$ be Bernoulli distribution random variables with a hit (success) probability P_{hit} and a miss (failure) probability $1 - P_{hit}$. Assume $y_j, j = 1,\ldots,T$ are independent identical distribution random variables. Then, $Y(T) = \sum_{j=1}^{T} y_j$ is a binomial distribution random variable with the parameter T and the hit (success) probability P_{hit}. The mean of $Y(T)$ is TP_{hit}, and the variance of $Y(T)$ is $TP_{hit}(1-P_{hit})$. According to the central limit theorem, the binomial random variable $Y(T)$ with a large T is approximated by a normal distribution. ■

Since $Y(T)$ represents the number of hits after T samples, $\hat{P}_{hit} = \frac{Y(T)}{T}$ is the maximum likelihood estimator for P_{hit}. The α percentile confidence interval is given by

$$\hat{P}_{hit} \pm Z_\alpha \sqrt{\frac{P_{hit}(1-P_{hit})}{T}} \quad (7)$$

Then

$$2 \cdot Z_\alpha \sqrt{\frac{P_{hit}(1-P_{hit})}{T}} = \beta \quad (8)$$

The maximum value of $P_{hit}(1-P_{hit})$ is 0.25 when $P_{hit} = 0.5$. In order to satisfy the accuracy requirement, the minimum sample size T is given by

$$T = \frac{Z_\alpha^2}{\beta^2} \quad (9)$$

After T packets are sampled, the estimate of $\hat{P}_{hit}$ for P_{hit} belongs to the interval $\hat{P}_{hit} \in (P_{hit} - \frac{\beta}{2}, P_{hit} + \frac{\beta}{2})$ with a probability greater than α.

4.2 The convergence of the hit probability for choosing k packets in SRED

For an incoming packet, the probability that the packet belongs to flow i is P_i. In the case of choosing k packets from the Zombie list and comparing all of them with the incoming packet, the probability that there is at least one of the k packets belonging to flow i is $P_{zi} = 1 - (1-P_i)^k$. Since there are tens of thousands of flows in the router, it is reasonable to assume that the probability P_i is generally small. Thus, the probability P_{zi} can be approximated by $P_{zi} \approx kP_i$. Furthermore, the hit probability that the incoming packet belongs to the flow i is kP_i^2. Then the hit probability for an incoming packet is equal to the sum of the hit probability for each flow:

$$P_{hitk} = \sum_{i=1}^{N} kP_i^2 = kP_{hit} \quad (10)$$

Theorem 2: Let $Y(T)$ represents the number of hits after T samples. For large T,

$$\sqrt{T}\,[\frac{Y(T)}{T} - P_{hitk}] \sim N[0, P_{hitk}(1-P_{hitk})] \quad (11)$$

Proof: Same as Theorem 1. ■

Theorem 3: Let X_n be a sequence of statistics such that

$$\sqrt{n}\,[\frac{X_n}{n} - \theta] \rightarrow X \sim N[0, \sigma^2(\theta)] \quad (12)$$

Let f be a differentiable function of one variable. Then

$$\sqrt{n}\,[f(\frac{X_n}{n}) - f(\theta)] \rightarrow f(X) \sim N[0, \sigma^2(\theta)(f'(\theta))^2] \quad (13)$$

Proof: See Rao [6] for the details of the proof. ■

In this case, θ is equal to P_{hitk}, and $f(\theta)$ is equal to P_{hit}. Then,

$$f(\theta) = \frac{1}{k}\theta \quad (14)$$

From Theorem 3,

$$\sqrt{T}\,[\frac{Y(T)}{k \cdot T} - P_{hit}] \sim N[0, \frac{1}{k^2} k\,P_{hit}(1-kP_{hit})] \quad (15)$$

Since $Y(T)$ represents the number of hits after T samples, $\hat{P}_{hit} = \frac{Y(T)}{kT}$ is the maximum likelihood estimator for P_{hit}. The α percentile confidence interval is given by

$$\hat{P}_{hit} \pm Z_\alpha \sqrt{\frac{1}{k^2}\frac{kP_{hit}(1-kP_{hit})}{T}} \quad (16)$$

Then

$$2 \cdot Z_\alpha \frac{1}{k} \sqrt{\frac{kP_{hit}(1-kP_{hit})}{T}} = \beta \quad (17)$$

The maximum value of $kP_{hit}(1-kP_{hit})$ is 0.25 when $P_{hit} = \frac{1}{2k}$. In order to satisfy the accuracy requirement, the minimum sample size T is given by

$$T = \frac{Z_\alpha^2}{k^2\beta^2} \quad (18)$$

To achieve the same estimated accuracy of the hit probability, the sample size of choosing k packets is $\frac{1}{k^2}$ of the sample size of choosing one packet for each incoming packet. In other words, although the convergence speed is not increased by k^2 due to nonlinearity, we can say that the convergence speed is increased in the usual cases.

4.3 Speeding up the convergence of estimated number of flows in CSFQ

To speed up the convergence, a factor k is multiplied with the probability $\hat{P}_i$ to emulate choosing k packets to compare with the incoming packet. When we sum the probability $k\hat{P}_i$ for all n packets without classifying them into the different flows, we can get a sum m:

$$m = nkP_1^2 + nkP_2^2 + \cdots + nkP_N^2 = nk\sum_{i=1}^{N} P_i^2 \quad (19)$$

Then,

$$P_{hit} = \sum_{i=1}^{N} P_i^2 = \frac{m}{nk} \quad (20)$$

However, if the factor k is multiplied with the probability $\hat{P}_i$ for each packet, the sum m is divisible by the factor k. The factor k will be canceled in the numerator and the denominator.

Let us carefully study the case of choosing k packets in SRED. If the incoming packet belongs to flow i and the probability P_i is small, the probability that there is at least one packet of the k packets belonging to flow i is approximately kP_i. Thus, the number of hits for flow i is approximately increased by k. On the other hand, if the probability P_i is large, the probability that there is at least one packet of the k packets belonging to flow i is not increased by k since the probability is not increased linearly. Thus, the number of hits for flow i is not increased by k.

Therefore, the incoming packets are roughly classified into two classes based on the packet label and the estimated fair share. If the incoming packet label is less than the estimated fair share, we consider the real packet as k virtual packets with the same packet label. It means that $\hat{P}_i$ multiplies a factor k, and the number of incoming packets (In fact, the number of virtually incoming packets) is added by k . On the other hand, if the incoming packet label is greater than the estimated fair share, we consider the real packet as k/x ($x = 2$ or 3 empirically) virtual packets with the same packet label. The factor x depends on the packet label and the estimated fair share. Further research includes finding the proper factor x for each incoming packet. The detailed pseudo code is given in the next section.

Let us see the example again. The factor k is equal to 3. The estimated number of flows converges to 505 after 5 iterations. Figure 2 gives the convergence process of the estimate. The convergence process speeds up.

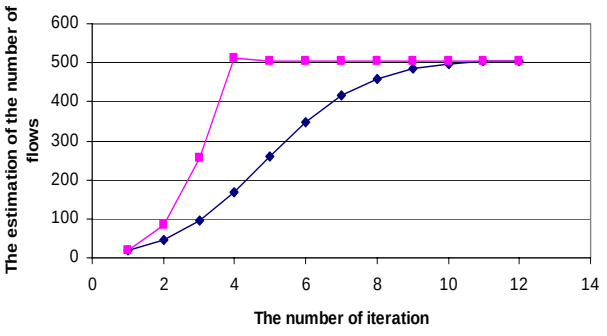

Figure 2. The convergence of the estimated number of flows

```
Estimate_α ( p, dropped)      // α is the estimated fair share.
  // α is initialized to the packet label when the queue size
  // first reaches the threshold (1/4 of the buffer size)
  A  = estimate_rate ( A , p); //estimate aggregate arrival rate
  if ( dropped == TRUE)
          return;
  if ( A > C)                          //congestion
    if ( pkt_label < α)
          m = m + k * pkt_label / A;
          n_last_step = n_last_step + k;
    else
          m = m + k/3 * pkt_label / A;
          n_last_step = n_last_step + k/3;
  else                                 //uncongestion
     m = m +  pkt_label / A;
  n++;                                 // enqueueing number of packets
  if ( n == 1./5*SAMPLE_SIZE)
          N = ( n_last_step + n_last)/ ( m + m_last);
  if ( n == SAMPLE_SIZE)
          n_last = n_last_step; //update n_last and m_last
          m_last = m;
          n = 0; n_last_step = 0; //initialize n, m, n_last_step for
          m = 0;                       // the next sampling period
return α = C / N;
```

Figure 3. Pseudo of fair share estimate

4.4 Implementation of the Speedup Method

Let us define n as the sampling size and the time to accept these n packets as a sampling period. After n packets are accepted by the router, the estimated fair share bandwidth is updated. In this paper the sampling size n is defined as the number of packets transmitted successfully in the output link for a 0.1s interval. Since the packet size is fixed at 1000 bytes and the link bandwidth is $10M$bps, the sampling size n is about $1,000$ in the simulations. Further research includes determining n from T appeared in the sections 4.1 and 4.2 to achieve a better performance.

During the congestion time $(A > C)$, the speedup method is used. If the incoming packet label is less than the estimated fair share, $\hat{P}_i$ multiplies a factor k and the number of virtually incoming packets is increased by k. If the incoming packet label is greater than the estimated fair share, $\hat{P}_i$ multiplies a factor $k/3$ and the number of virtually incoming packets is increased by $k/3$. The factor k may take the value from 3 to 15 empirically. During the uncongestion time $(A \leq C)$, the incoming packet is equal to one virtually incoming packet. Further research includes choosing the proper value k and evaluating the performance of choosing different k.

However, setting a constant estimated fair share for the current sampling period may result in an oscillation of the throughput of the flow. To solve the oscillation problem, BLACK (for BLACKlist unresponsive flows) [7] gives hints that the n and m in the previous sampling period need to be considered to improve the estimated accuracy. Therefore, we further divide the sampling size into five equal length segments. In the current sampling period, when the

number of the really accepted packets is q ($q = 1, 2, 3, 4, 5$) times the segment size, we update the fair share based on the information from both the current and previous sampling periods. The equation (21) is used to update the number of flows:

$$N = \frac{n\_last\_step + n\_last}{m + m\_last} \tag{21}$$

where m_last and n_last are the value of m and the number of virtually accepted packets in the last sampling period respectively; m and n_last_step are the accumulated sum of $\hat{P}_i$ (in fact $k\hat{P}$ or $\frac{k}{3}\hat{P}$ or $\hat{P}$) and the number of virtually accepted packets in the current period respectively.

When the number of the really accepted packets n is equal to the sampling size, the value of m_last and n_last are updated to the m and n_last_step of the current period. This modification responds quickly to changing dynamics in the flow rates. The pseudo code reflecting this algorithm is described in Figure 3.

5 Simulations

In this section we evaluate our proposal using the ns-2 simulator. The speedup method is compared with the original heuristic method and our previous probabilistic method in a series of experiments. We call the probabilistic method as CSFQIMP and call the speedup method as CSFQIMPK. We compare the estimated fair share of the three methods in the following experiments. Due to the space limitation, we are unable to show all our simulation evaluations.

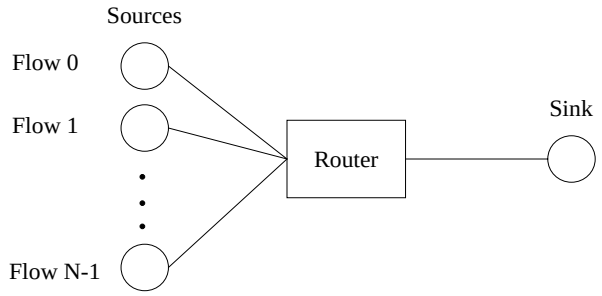

Figure 4. Single congested link

The simulation set-up used in this paper is identical to that in reference [1]. Unless otherwise specified, we use the same parameters as those in CSFQ. Each output link has a latency of 1ms, a buffer of 64kB, and a buffer threshold is 16kB. The averaging constant used in estimating the flow rate is $K = 100$ms. The packet size is fixed at 1k bytes, and the simulation time is 10s. The sampling size n is $1,000$. The factor k is 9. Detailed descriptions of other simulation parameters of CSFQ can be found in [1].

5.1 Single Congested Link

The topology of the first set of simulations is shown in Figure 4. The single congested link is shared by N flows, and we evaluate the estimated fair share with two experiments.

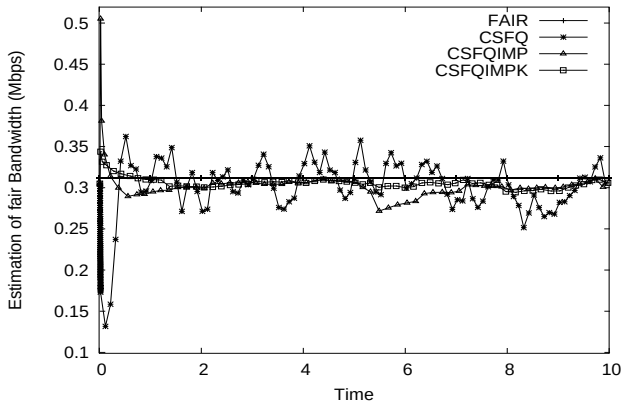

Figure 5. Fair share estimation

In the first experiment, 32 CBR (Constant Bit Rate) flows, indexed from 0, share the 10Mbps bottleneck link. The flow rate of flow i is $(i + 1)$ times more than its fair share, i.e. $(i+1)\cdot 10/32$Mbps. Figure 5 shows the estimated fair share over a $10-$s interval. CSFQIMPK performs better than CSFQIMP. It converges faster and estimates more accurate than CSFQIMP.

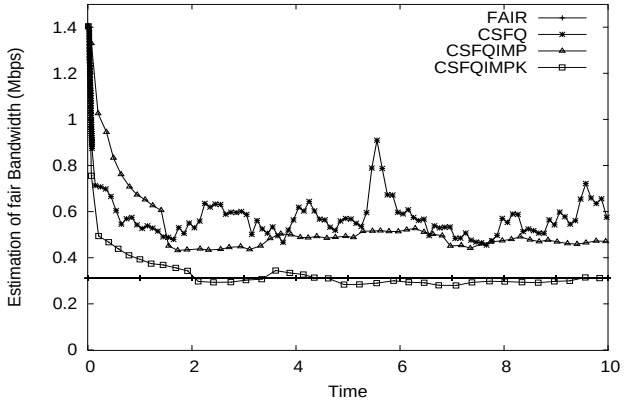

Figure 6. Fair share estimation

In the second experiment, we evaluate the impact of an CBR flow (Flow ID $= 0$) on a set of 31 TCP flows. The flow rate of the CBR flow is 10Mbps, which tries to occupy all of the link capacity. Figure 6 shows the estimated fair share over a $10-$s interval. CSFQIMPK converges much faster and more accurate than CSFQIMP.

5.2 Multiple Congested Links

The second set of simulations is run with the topology shown in Figure 7. The purpose is to analyze how the throughput of a well-behaved flow is affected when the flow traverses more than one congested link. Due to the space limitation, we only show the case that the well-behaved flow is a TCP flow and there are 5 congested links. The

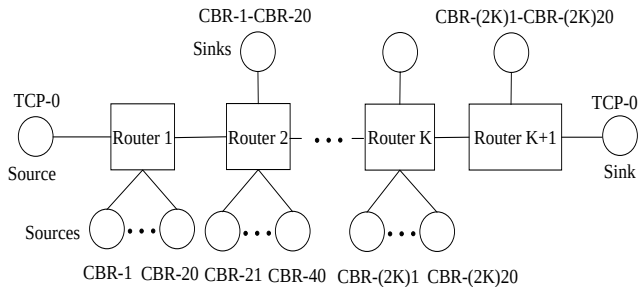

Figure 7. Multiple congested links

cross traffic enters the path in one of the routers and exits at the next. The CBR sources that formed the cross traffic are now replaced with ON/OFF sources. The burst (ON) and idle (OFF) time periods are both exponentially distributed with the same average 0.5 sec. Figure 8 shows the estimated fair share of each congested link for CSFQ, CSFQIMP, and CSFQIMPK respectively. As expected, CSFQIMPK is better than CSFQIMP.

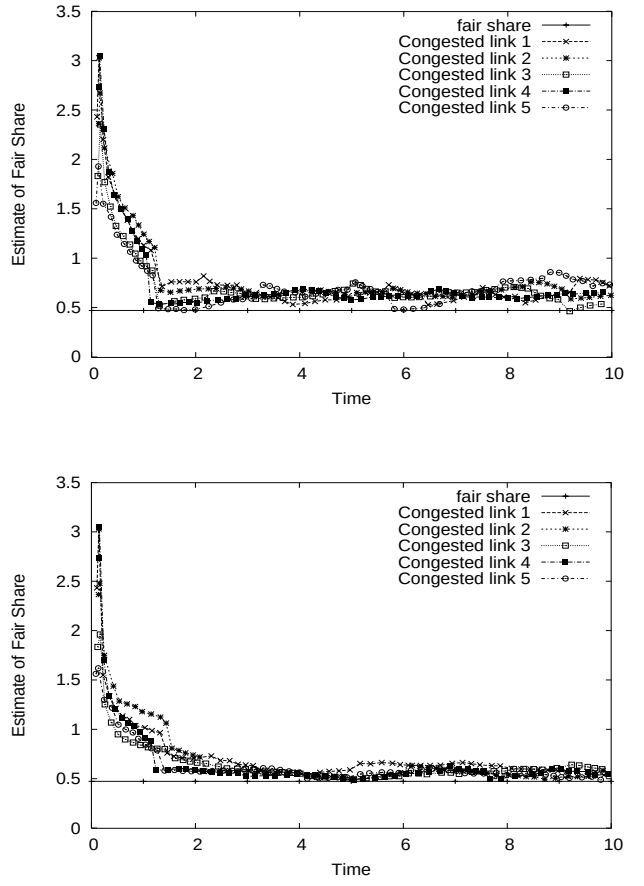

Figure 8. (a) Fair share estimate of CSFQIMP for 5 congested links (b) Fair share estimate of CSFQIMPK for 5 congested links

6 Conclusion

In the paper [3] we propose a probabilistic approach for achieving fair bandwidth allocations in CSFQ. However, the convergence speed of the estimate is slow, which makes the stationary assumption suspect. Thus, there are a number of situations that the probabilistic method cannot handle well due to an inaccuracy in estimating the number of flows.

In this paper we propose a method to speed up the convergence of estimating the number of flows. The idea comes from that the router randomly selects k packets instead of one packet to compare with the incoming packet in SRED. We discuss its design goals and present the performance simulations and experiments that demonstrate its performance compared to the existing scheme in various scenarios. The speedup approach achieves satisfactory performance. Further research includes determining the speedup factor k/x for each incoming packet based on the flow rate and the estimated fair share.

It is interesting to note that TCP flows are difficult to achieve the fair share bandwidth in CSFQ due to the TCP's congestion control mechanism. Further research includes developing a rate-based transport protocol that can achieve the fairness where the routers are responsible for estimating the fair share and allocating the bandwidth.

References

[1] I. Stoica, S. Shenker, and H. Zhang, Core-stateless fair queueing: a scalable architecture to approximate fair bandwidth allocations in high-speed networks, *IEEE/ACM Transactions on Networking*, 11(1), 2003, 33-46.

[2] T. J. Ott, T. V. Lakshman, and L. H. Wong, SRED: Stabilized RED, *Proc. of IEEE INFOCOM'99*, New York, USA, 1999, 1346-1355.

[3] P. Wang and D. Mills, A probabilistic approach for achieving fair bandwidth allocations in CSFQ, *Proc. of IEEE NCA'05*, Cambridge, MA, USA, 2005, Accepted.

[4] D. Bertsekas and R. Gallager, *Data Networks* (New York: Prentice Hall, 1987).

[5] M. Kodialam, T. V. Lakshman, and S. Mohanty, Runs based traffic estimator (RATE): A simple, memory efficient scheme for per-flow rate estimation, *Proc. of IEEE INFOCOM'04*, Hong Kong, China, 2004.

[6] C. R. Rao, *Linear Statistical Inference and its Applications* (New York: John Wiley and Sons Inc., 1973).

[7] M. Labrador, and S. Banerjee, Black: Detection and preferential dropping of high bandwidth unresponsive flows, *Proc. of IEEE ICC'03*, Anchorage, Alaska, USA, 2003, 664-668.