A Probabilistic Approach for Achieving Fair Bandwidth Allocations in CSFQ

Peng Wang David L. Mills
Department of Electrical & Computer Engineering
University of Delaware
Newark, DE 19716
pwangee@udel.edu; mills@eecis.udel.edu

Abstract

The fair bandwidth allocations can isolate flows and protect well-behaved flows from ill-behaved ones. CSFQ (Core Stateless Fair Queueing) achieves the approximate fairness by dropping the extra packets beyond the fair share bandwidth at the routers. A heuristic method is used to estimate the fair share in CSFQ. Furthermore, we know that SRED (Stabilized RED) uses a probabilistic method based on a Zombie list to estimate the number of flows at the router. In this paper, we take the probabilistic idea from SRED and apply it in CSFQ to estimate the fair share without using the Zombie list. Simulation results show that the new probabilistic approach achieves a comparable or even better performance than the original heuristic approach.

1. Introduction

The current Internet provides a connectionless, best-effort, and end-to-end packet service by using the IP protocol. The stability of the Internet architecture depends to a large extent on the end-to-end congestion control mechanisms provided by TCP. However, many implementations of TCP do not include congestion control mechanisms either deliberately or by accident. Moreover, more and more UDP-based applications appear on the Internet, such as video and audio. The misbehaving flows can consume most of the bandwidth and starve out TCP-friendly flows.

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 IP packet header. In CSFQ all routers are classified as edge routers or core routers. The edge routers maintain the per-flow state, and the core routers are stateless. The edge routers measure the per-flow rate and attach the flow rate as a label to each packet header. All routers (including edge routers and 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. From extensive simulations, CSFQ approaches the fair share of DRR (Deficit Round Robin) [6] and outperforms FIFO (First In First Out), and RED (Random Early Detection) [4].

In order to stabilize its buffer occupation, SRED [5] determines the dropping probability for each incoming packet based on the estimated number of flows. A probabilistic method based on a Zombie list is used to estimate the number of flows at the router. A Zombie list consists of a shift register that is much larger than the router's queue, and stores the recently seen packets. For an incoming packet, a packet is randomly selected from the Zombie list and compared with the incoming packet. If they are from the same flow, it is called a "hit". Otherwise, it is called a "miss". The number of incoming packets divided by the number of hits is a good estimate for the effective number of active flows in the network. Furthermore, the estimated fair share is the link bandwidth divided by the estimated number of flows.

In this paper, we take the probabilistic idea from SRED and apply it in CSFQ 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 new probabilistic approach achieves a comparable performance to the original heuristic approach. Extensive simulations are used to show the performance of our probabilistic method.

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. In Section 4, we evaluate

[1]This research is sponsored by the NSF grant ANI-0312851. 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 NSF or the US goverment.

the performance of our method in comparison to the original CSFQ by using simulations. Finally, we conclude in Section 5.

2. CSFQ Architecture

2.1. Objectives

The primary objective is to achieve max-min fairness [1] among the flows in a congested router. This qualitative property can be summarized as follows:

> We allocate bandwidth max-min fairly if it is not possible to increase the satisfaction of a flow without simultaneously causing the decrease in the satisfaction of a less satisfied flow.[3]

In fact, max-min fair bandwidth allocations are characterized by the fact that all flows that are bottlenecked by this router have the same output rate. We call this rate the fair share rate of the link.

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 $r_i(t)$ are estimated at the edge routers, and the estimated flow rates 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 [7].

Figure 1 shows that a FSM (finite state machine) with three states is used to describe how to estimate the fair share. Three states are defined:

- Congested: $A > C$ at all times during a time interval of length K_c
- Uncongested: $A < C$ at all times during a time interval of length K_c
- Normal: Otherwise

where C is the link capacity and K_c is a window size to filter out the inaccuracy due to the exponential smoothing.

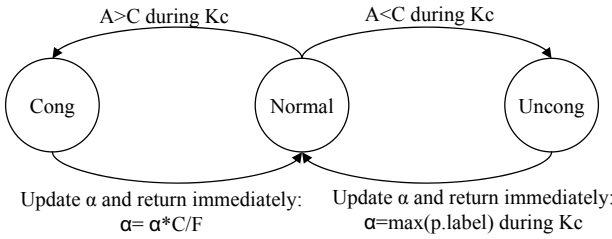

Figure 1. FSM for fair share estimate in CSFQ

State Normal is a permanent state, while state Congested and state Uncongested are both transient states. If the condition for state Uncongested is satisfied, the fair share α is set to the largest rate of the active flows in the last K_c time units and the link state returns to state Normal immediately. If the condition for state Congested is satisfied, the fair share rate α is updated by the formula $\alpha_{new} = \alpha_{old} \cdot C/F$, and the link state returns to state Normal immediately too. Generally, the fair share α is updated at the end of the interval K_c no matter for state Congested or state Uncongested.

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

To achieve fair bandwidth allocations, the easy way is to know the number of flows passing through the router and then allocate the bandwidth among the flows evenly. However, it is impossible to know the exact number of flows if the router does not maintain the per-flow state. A probabilistic method to estimate the number of flows is used in the algorithm SRED without maintaining the per-flow state.

3.1. Estimate the Number of Flows in SRED

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. We also assume that the probability that an arriving packet belongs to a given flow is independent of all other packets. For an incoming packet, the probability that the packet belongs to flow i is P_i. The probability that a randomly selected packet from the Zombie list belongs to flow i is P_i too, since the Zombie list is large. Thus, the hit probability that each arrival packet belongs to flow i is P_i^2. Then the hit probability for an incoming packet is equal to the sum of the hit probability for each flow:

$$P_{hit} = \sum_{i=1}^{N} P_i^2 \tag{3}$$

In general, the hit probability satisfies the equation:

$$\frac{1}{N} < P_{hit} < 1 \tag{4}$$

The lower limit is achieved only when N flows have the same traffic intensity. The upper limit is achieved only when one flow sends all packets and the other $(N-1)$ flows send zero packets. 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.

$$n \cdot \frac{1}{N} \approx n \cdot P_{hit} = m \tag{5}$$

The estimated hit probability is m/n, and the estimated number of flows is equal to n/m. Furthermore, the estimated fair share is equal to $C \cdot m/n$ where C is the output link capacity. We realize that the estimated number of flows is always less than the actual number of flows for the asymmetric traffic since $n \cdot \frac{1}{N} < n \cdot P_{hit} = m$, and then $N > n/m$.

3.2. Estimate the Number of Flows in CSFQ

We take the probabilistic idea from SRED and apply it to estimate the number of flows in CSFQ without the Zombie list. The algorithm and the implementation are described in the following two sections.

3.2.1 Estimate the Number of Flows

We assume that each arrival packet belongs to one of the N flows. 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. We also assume that the probability that an arriving packet belongs to a given flow is independent of all other packets.

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 into the different flows, we can get a sum m:

$$m = nP_1^2 + nP_2^2 + \cdots + nP_N^2 = n\sum_{i=1}^{N} P_i^2 = nP_{hit} \tag{6}$$

This is the same as the formula (5) used in SRED. The estimated number of flows is equal to n/m. Furthermore, the

estimated fair share is equal to $C \cdot m/n$ where C is the output link bandwidth.

One of the problems encountered by the above method is an inaccuracy when the traffic density is vastly different. The heavy flows with large labels contribute more than the light flows to the value m and affect the estimated accuracy of the fair share. Thus, we consider n accepted packets instead of n incoming packets at the routers. The incoming packets are classified as successfully transmitted packets, dropped packets due to dropping policy and dropped packets due to queue overflows. The accepted packets include successfully transmitted packets and dropped packets due to the queue overflow at the routers. Not surprising, most dropped packets are from the heavy flows. Therefore, 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.

At the end of the first second, $5,000$ packets from the first aggregate flow and $5,000$ packets from the second aggregate flow are accepted by the core router (although $5,000$ packets are dropped due to the queue overflows). The m is equal to $5000 \cdot 1/1000 + 5000 \cdot 1/10 = 505$. Thus, the estimated number of flows is $N = 10000/505 = 19.8020$, and the fair share is $C/N = 5000/19.8020 = 252.4997$.

At the end of the second second, $5,000$ packets from the first aggregate flow and $5 \cdot 252.4997$ packets from the second aggregate flow are accepted by the core router. The m is equal to $5000 \cdot 1/1000 + 5 \cdot 252.4997 \cdot 1/10 = 131.2498$. Thus, the estimated number of flows is $N = (5000 + 5 * 252.4997)/131.2498 = 47.7143$, and the fair share is $C/N = 5000/47.7143 = 104.7904$.

The estimated number of flows converges to 505 after 12 iterations. Figure 2 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 may be slow.

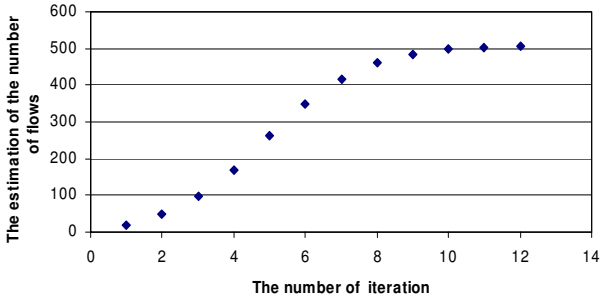

Figure 2. The convergence of the estimated number of flows

3.2.2 Implementation of the Probabilistic 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, and n, m are reset. We need to determine the value of n that affects the estimated accuracy. 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. In the following simulation section, the link bandwidth is 10Mbps, and the packet size is fixed at 1000 bytes. Thus, the sampling size n is about $1,000$.

However, setting a constant estimated fair share for the next sampling period may result in a oscillation of the throughput of the flow. To solve the oscillation problem, BLACK (for BLACKlist unresponsive flows) [2] gives hints that the n and m in the current sampling period need to be considered to improve the estimated accuracy. Therefore, we further divide the sampling size into five equal length segments, and the segment size is $1/5$ of the sampling size. In the current sampling period, when the number of the accepted packets is $k(k = 1, 2, 3, 4, 5)$ times the segment size, we update the fair share based on the information from both the current and previous sampling periods. The equation (7) is used to update the number of flows:

$$N = \frac{n + n\_last}{m + m\_last} \tag{7}$$

where m_last is the value of m in the last sampling period, n_last is the sampling size, m is the accumulated sum of $\hat{P}_i$ in the current sampling period, and n is the number of the accepted packets in the current period. When the number of the accepted packets is equal to the sampling size, the value of m_last and n_last are updated to the m and n 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. We further reduce the impact of the heavy traffic by multiplying coefficients to $\hat{P}_i$. If the current queue size is larger than the threshold and the packet label is greater than

```
Estimate_α ( p, dropped)
  // α is initialized to the packet label when the queue size
  // first reaches the threshold
 A = estimate_rate (A , p); //estimate aggregate arrival rate
 if ( dropped == TRUE)
                return;
 if (A>C)
    if ( pkt_label > α && queue_size > Threshold)
                m = m + 0.75 * pkt_label /A;
    else if ( pkt_label > α)
                m = m + 0.98 * pkt_label /A;
    else
                m = m + 1 * pkt_label /A;
 else
                m = m + 1 * pkt_label /A;
 n++; // enqueueing number of flows
 if ( n == 1./5 * SAMPLE_SIZE)
                N = ( n + n_last) / ( m + m_last) ;
 if ( n == SAMPLE_SIZE)
                n_last = SAMPLE_SIZE;
                m_last = m;
                n = 0;
                m = 0;
return α = C/N;
```

Figure 3. Pseudo of fair share estimate

the estimated fair share, $\hat{P}_i$ multiplies a coefficient 0.75. If the packet label is greater than the estimated fair share, $\hat{P}_i$ multiplies a coefficient 0.98. These two coefficients are given arbitrarily and may be varied in a range without affecting the performance greatly.

4. Simulations

In this section we evaluate our proposal using the ns-2 simulator. Stoica [7] has shown using simulations that CSFQ achieves a good performance when compared with FIFO, RED, FRED (Flow Random Early Drop) and DRR. Thus, we only compare our modified CSFQ with the original CSFQ in a series of experiments to see whether the probabilistic estimation approach for the estimated fair share is accurate and gives a comparable performance. We call the modified CSFQ as CSFQIMP. 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 $1,000$. Detailed descriptions of other simulation parameters of CSFQ can be found in [7].

4.1. 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 fairness among the flows with three experiments.

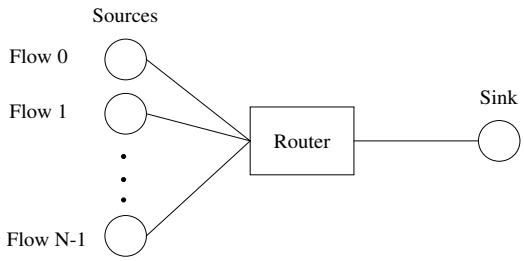

Figure 4. Single congested link

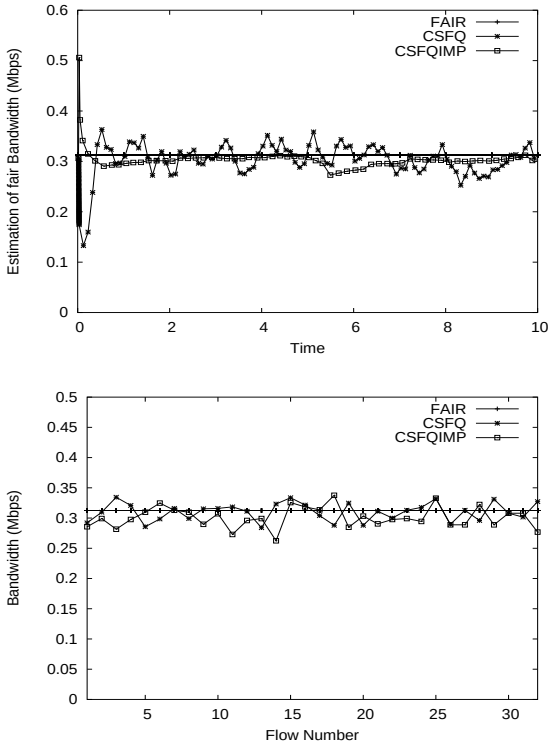

Figure 5. (a) Fair share estimation (b) Average throughput over 10s

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(a) shows the estimated fair share over a $10-$s interval. Both of the estimates of the fair share are comparable. In the probabilistic method, the estimated fair share should converge to the actual fair share, since the flow rates of CBR flows are stationary. But in some portion of the graph, the estimated fair share is less than the theoretical value. The reason is that we multiply a coefficient 0.75 to reduce the impact of the heavy flows. Figure 5(b) shows the average throughput

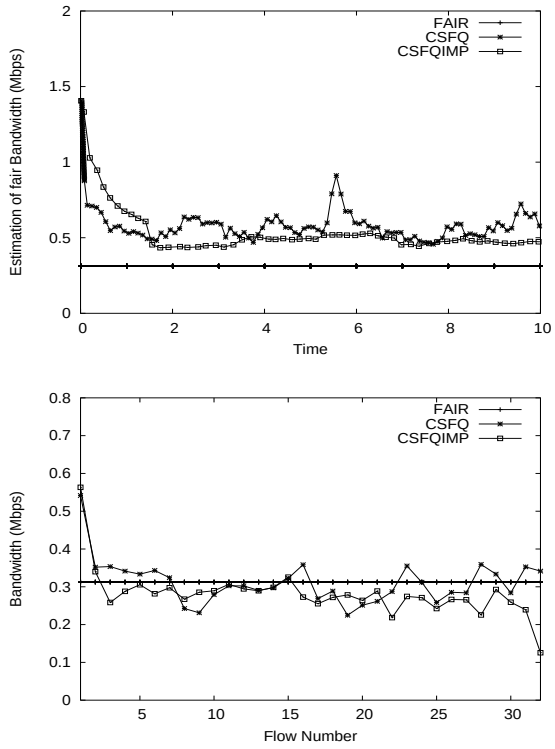

Figure 6. (a) Fair share estimation (b) Average throughput over 10s

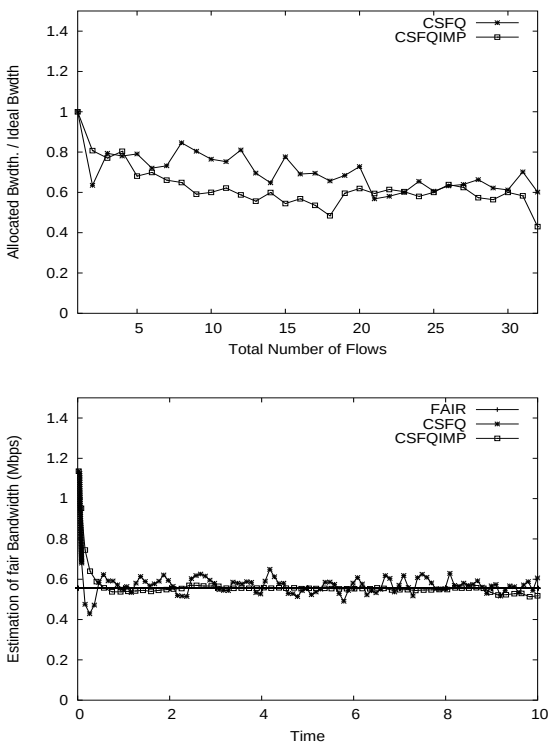

Figure 7. (a) A TCP flow sharing the link with N CBR flows (b) Fair Share Estimate for 18 flows

of each flow over a $10-$s interval. CSFQ and CSFQIMP achieve a comparable performance.

In the second experiment, we evaluate the impact of an ill-behaved 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(a) shows the estimated fair share over a $10-$s interval. The probabilistic method converges slower but is more accurate than the heuristic method. Figure 6(b) shows the average throughput of each flow over a $10-$s interval. The performance of CSFQIMP is a little worse than that of CSFQ, although the estimated fair share in CSFQIMP is more accurate. The reason is that the congestion control mechanism of TCP reduces the transmission window size to half when the flow rate reaches the estimated fair share. For example, if there are 15 consecutive packets with the flow rate 10% over the estimated fair share, the probability that at least one packet is dropped is $1-(1-0.1)^{15} = 79.41\%$. Furthermore, the more accurate the estimated fair share is, the worse is the TCP performance due to the congestion control mechanism in this scenario.

In the third experiment, we evaluate a TCP flow against an increasing number of ill-behaved CBR flows. We perform 31 simulations. In the ith simulation, one TCP flow is against i CBR flows. The CBR flow rate is twice the fair share rate. Figure 7(a) shows the normalized bandwidth of TCP flow that competes with N CBR flows. The performance of CSFQIMP is a little worse than that of CSFQ. The reason is also due to the congestion control mechanism of TCP. Figure 7(b) shows the estimated fair share in the case with 17 CBR flows. The estimated fair share in CSFQIMP is comparable to, or even better than that in CSFQ.

4.2. Multiple Congested Links

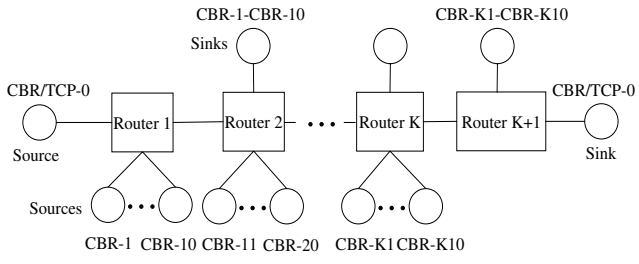

Figure 8. Multiple congested links

The second set of simulations is run with the topology shown in Figure 8. The purpose is to analyze how the

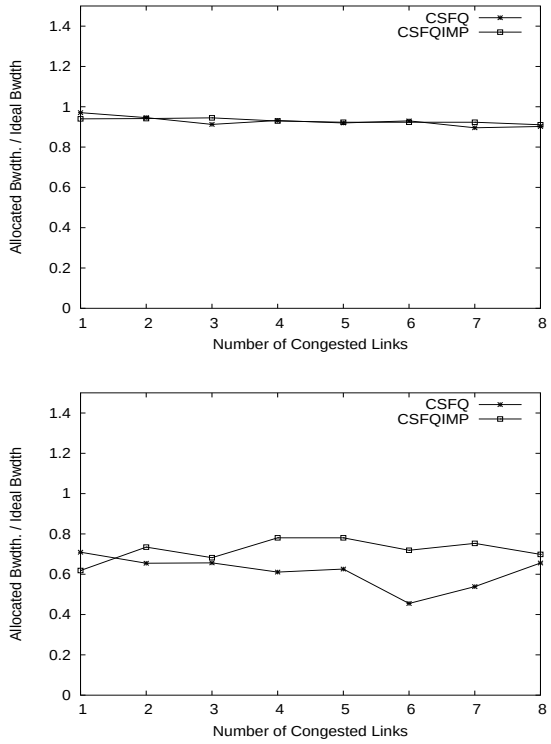

Figure 9. (a) Normalized throughput of CBR-0 as a function of the number of congested links (b) CBR-0 is replaced by a TCP flow

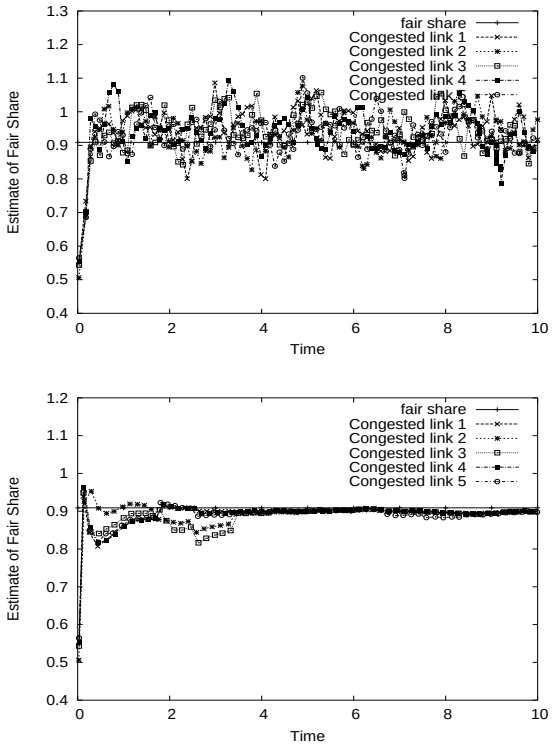

Figure 10. (a) Fair share estimate of CSFQ for 5 congested links (b) Fair share estimate of CSFQIMP for 5 congested links

throughput of a well-behaved flow is affected when the flow traverses more than one congested link. All CBR flows, except CBR-0, send at 2Mbps. The cross traffic enters the path in one of the routers and exits at the next. The well-behaved flow is a UDP flow sending at its fair share or a TCP flow.

In the first experiment, the well-behaved flow is CBR flow (CBR-0) sending at its fair share rate of 0.909Mbps. Figure 9(a) shows the normalized bandwidth of CBR-0 versus the number of the congested links. The UDP flow is not much affected by the cross traffic and achieves a good fairness in both CSFQ and CSFQIMP.

In the second experiment, a TCP flow replaces the CBR-0. Figure 9(b) shows the normalized bandwidth of TCP versus the number of the congested links. CSFQIMP performs better than CSFQ since the cross CBR traffic is reduced by the accurate estimate of the fair share in CSFQIMP. In the case with 5 congested links, the estimated fair share of the heuristic method is given in the Figure 10(a), and the estimated fair share of the probabilistic method is given in Figure 10(b). We plot the estimated fair share for each congested link. As expected, the probabilistic method is better than the heuristic method in this case.

4.3. Bursty cross traffic

The simulations are run with the topology shown in Figure 8 with 5 congested links. 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 chosen between 5 msec and 0.5 sec.

In the first experiment, the well-behaved flow is CBR flow (CBR-0) sending at its fair share rate of 0.909Mbps. Figure 11(a) shows the normalized bandwidth of CBR-0 versus the burst time period. The UDP flow is not much affected by the bursty cross traffic and achieves a good fairness in both CSFQ and CSFQIMP.

In the second experiment, a TCP flow replaces the CBR-0. Figure 11(b) shows the normalized bandwidth of TCP versus the burst time period. CSFQIMP performs better than CSFQ. In the case with 0.5s average burst time, the estimated fair share for both methods are given in the Figure 12(a) and 12(b). We plot the estimated fair share for each congested link. As expected, the probabilistic method is better than the heuristic method in this case.

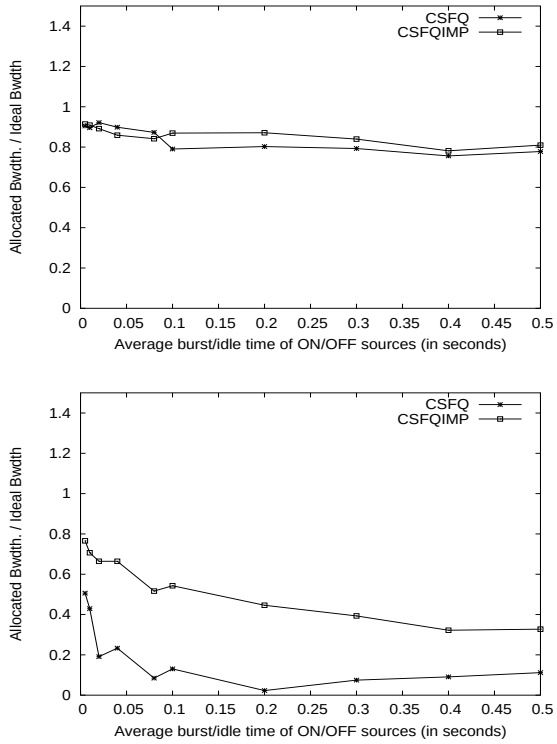

Figure 11. (a) Normalized throughput of CBR-0 as a function of the number of congested links (b) CBR-0 is replaced by a TCP flow

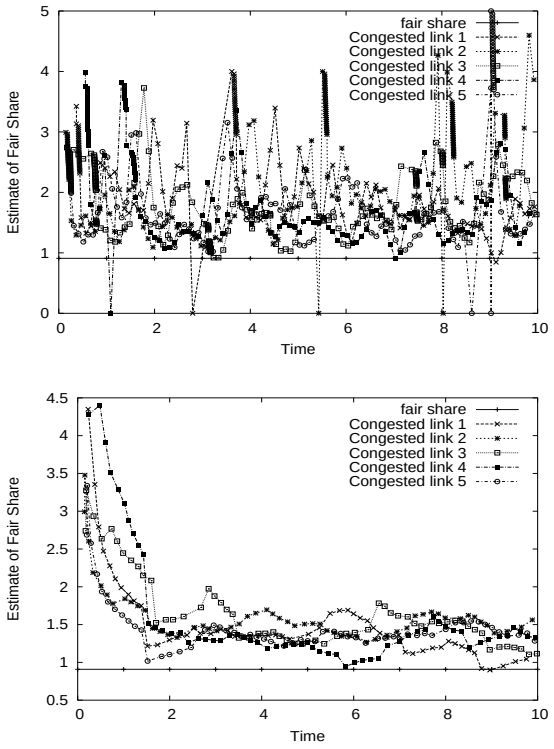

Figure 12. (a) Fair share estimate of CSFQ for 5 congested links (b) Fair share estimate of CSFQIMP for 5 congested links

5. Conclusion

In this paper we present a probabilistic approach for achieving fair bandwidth allocations in CSFQ. The probabilistic idea is taken from SRED and applied in CSFQ without using the Zombie list. 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 probabilistic method is simple and easy to implement. The estimated number of flows converges to the actual number of flows under the assumption that $\hat{P}_i$ is stationary. The performance of CSFQIMP is comparable to that of 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. Further research includes developing a method to speed up the convergence of the estimate.

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. Thus, although it is important to the stability of the current Internet, TCP congestion control mechanism may be poison in the scenario where the routers are responsible for allocating bandwidth. From the good performance of the UDP flows, a rate-based transport protocol may be more efficient than TCP in CSFQ.

References

[1] D. Bertsekas and R. Gallager. *Data Networks*. pp. 524-529, Prentice-Hall, 1987.

[2] G. Chatranon, M. Labrador, and S. Banerjee. Black: Detection and preferential dropping of high bandwidth unresponsive flows. In *Proceedings of IEEE ICC*, pages 664–668, May 2003.

[3] A. Clerget and W. Dabbous. TUF : Tag-based unified fairness. In *Proceedings of INFOCOM*, pages 498–507, 2001.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[5] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, volume 3, pages 1346–1355, 1999.

[6] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *Proceedings of ACM SIGCOMM'1995*, pages 231–243, 1995.

[7] 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*, 11:33–46, February 2003.