A Credit-based Active Queue Management (AQM) Mechanism to Achieve Fairness in the Internet

Gwyn Chatranon[1], Miguel A. Labrador[2], and Sujata Banerjee[3]

[1] University of Pittsburgh gwync@mail.sis.pitt.edu
[2] University of South Florida labrador@csee.usf.edu
[3] Hewlett-Packard Laboratories sujata.banerjee@hp.com

Abstract. Router-based algorithms to address the TCP-friendliness problem have focused on providing fairness to TCP connections by penalizing UDP unresponsive flows. As a result, current schemes have overlooked their effect on streaming applications. In addition, current schemes have not been widely implemented in practice because of their high complexity and their inability to provide fairness when multiple unresponsive flows, packets of different sizes, and bursty traffic are present. All these aspects and scenarios, commonly found in practice, are rarely addressed in the literature. In this paper, we present *Achieving Fairness using a Credit-based mechanism* or AFC, a simple Fair Active Queue Management (FAQM) mechanism that aims to solve the unfairness problems generated under these realistic conditions. Simulation results show that AFC provides smoother transfer rates for unresponsive flows transmitting real-time traffic, handles multiple heavy unresponsive flows better, improves the fairness among TCP connections with different round-trip delays, and achieves good fairness even under bursty traffic and packets of different sizes.

Keyword: Active Queue Management, TCP-friendliness.

1 Introduction

Today, the majority of the Internet traffic such as web traffic, FTP, and email traffic, is carried by the TCP protocol. TCP is widely accepted because of the success of its congestion control mechanism, which enables end hosts to cooperatively adjust their transmission rates according to network conditions, and thus share the available bandwidth fairly among large number of users. Voice and video applications, on the other hand, utilize the UDP protocol. These applications, which are increasing in popularity over the Internet, send *unresponsive traffic* because the UDP protocol provides no end-to-end congestion control. As a result, when TCP and UDP-based applications share the same bottleneck link, the TCP-friendliness problem may arise [1].

Recently, a wide variety of applications and congestion control mechanisms have emerged, which may create an uncooperative environment for end hosts. As a result, leaving the Internet relying purely on end-host mechanisms is a potential risk on network performance and stability. Consequently, router-based mechanisms to address the fairness problem have been widely investigated during the past several years.

One important problem in existing FAQM schemes is that they have primarily focused their attention on providing fairness to TCP connections by penalizing UDP unresponsive flows. As a result, these schemes have overlooked the effect they produce on UDP applications. In addition, current schemes still present some fairness problems due to inaccuracies in the estimation of the number of active flows, innacuracies when multiple unresponsive flows are present, and unfairness when packets are of different sizes and traffic is bursty. All these problems, not analyzed thus far, are the main motivation for the new FAQM scheme proposed here. In addition, CARE [2], which is the newest scheme based on the capture-recapture model, has never been evaluated and compared along with the other schemes.

In this paper, a router-based scheme called AFC (Achieving Fairness using a Credit-based mechanism)[4] is proposed to prevent the unfairness problem using a small amount of per-flow state information while addressing all of these new issues. Through simulations, we compare AFC with CHOKe [4], Stochastic Fair Blue (SFB) [5], BLACK [6], and CARE [2], and show that it provides superior performance. The rest of the paper is structured as follows. Section 3 describes in detail the design goals and algorithms of the AFC scheme. The AFC scheme is then evaluated and compared with other different schemes in Section 4. Finally, conclusions and future work are provided in Section 5.

2 Related Work

Several Fair Active Queue Management (FAQM) schemes have been proposed to provide a fair share of bandwidth to competing flows. These schemes maintain only partial state information, have low computational and space complexity, and need no cooperation from network devices to achieve long-term fairness. In this paper we include CHOKe [4], Stochastic Fair Blue (SFB) [5], BLACK [6], and CARE [2].

CHOKe was introduced in [4] as a mechanism to provide fairness on top of a RED queue [7]. Upon each packet arrival, CHOKe randomly selects a packet from the queue, and drops both packets if they belong to the same flow. High bandwidth flows thus can be controlled as they usually have more packets in the buffer. However, simulations in [4] illustrate that, with CHOKe, a high bandwidth unresponsive flow still can gain much more bandwidth than the fair share. Besides, flows with larger packet sizes gain higher throughput. The only workaround suggested in [8] is to defrag the packets into smaller packets of the same size. Stochastic Fair Blue (SFB) [5] utilizes a bloom filter with multiple levels of hashing to detect high-bandwidth unresponsive flows probabilistically. However, SFB does not include a suggested mechanism to handle unresponsive flows except limiting them to a fixed amount of bandwidth. In practice, setting the bandwidth limit effectively is a difficult task as a fair bandwidth level may be unknown to an SFB router. Besides, if the number of unresponsive traffic is large enough, they might be hashed into the same locations as responsive traffic, causing a false detection of responsive traffic that would eventually be overpenalized. CARE [2], the most recent scheme, is based on the Capture-Recapture (CR) model that has been widely used to estimate the number of animals in a population and the number of defects in software

[4] AFC's credit-based mechanism has no relationship with a credit-based flow control in ATM network such as that appeared in [3].

inspection processes. This model is applied to estimate the flows' arrival rate and the number of active flows, and thus the fair share. Packets from unresponsive flows are dropped according to their proportion of arrival rates that exceed the fair share. The problem of CARE is its core mechanism to estimate the number of active flows. The algorithm is highly complex and presents problems if the intensity of the traffic are highly unequal [9]. Besides, the performance of CARE has never been evaluated along with the other schemes. BLACK, proposed in [6], uses the buffer occupancy fraction as an indicator of the flow's share of the bandwidth. Using a small cache memory, BLACK randomly samples a packet from the queue upon a packet arrival and updates the information in the cache memory to obtain only the high-bandwidth unresponsive flow candidates. BLACK uses a simple memory management adapted from the Least Recently Used (LRU) mechanism [10]. At the end of the sampling period, a buffer occupancy fraction, referred to as *HitFraction*, is estimated and the traffic that consume more than the fair share is penalized according to a dropping probability. In order to estimate the dropping probability p_{drop}, BLACK also estimates the number of active flows, where its inverse value becomes the $FairFraction$.

Even though BLACK has shown superior performance over CHOKe and SFB in [6], all of the fair AQM schemes listed above still contain some limitations. More importantly, these schemes have not yet been analyzed considering real networking situations and issues, such as the unfairness of the schemes due to inaccuracies in the estimation of the number of active flows, unfairness due to traffic with different packet sizes and different round-trip times, the performance of the schemes under bursty traffic conditions, and the fluctuation in the throughput of the flows after passing through the schemes. AFC, which was designed with all these issues in mind, is described next.

3 Achieving Fairness using a Credit-based Mechanism (AFC)

This section presents AFC, a new fair AQM scheme that addresses the limitations of the schemes listed in Section 2. AFC modifies BLACK in several ways with newly designed components to make AFC perform better than BLACK and the other schemes under realistic environments. At the same time, AFC inherits and improves over BLACK's important features of low computational and space complexity.

The first modification is related to the estimation of the number of active flows. BLACK presents performance problems because to calculate the number of active flows it assumes that the traffic intensity of the arriving flows is identical. In order to minimize possible estimation errors and to maintain a low level of complexity, in [9] we found that the Direct Bitmap method or any of its variants [11] is the best performing mechanism to estimate the number of active flows thus far. As a result, AFC estimates the number of active flow using the Direct Bitmap mechanism.

Most fair AQM schemes fail to provide fairness when flows send packets of different sizes. For example, CHOKe cannot prevent this problem without breaking a large packet into smaller packets of about the same size, as suggested by the authors in [8]. As CHOKe relies purely on packet matching, between a packet that is sampled from the queue and the arriving packet, a flow that has smaller packet size would be penalized more with the higher probability of matching. This is also the case of BLACK since it computes a flow's buffer fraction based on the number of packets of a particular flow over the number of total sampled packets. To solve this problem, instead of counting the number of packets for the candidate flows in the cache memory, the information is updated with the size of the sampled packet. In addition, the total number of bytes are counted in each period instead of the number of sampled packets. At the end of a sampling period, the *HitFraction* of a flow is calculated by the flow's byte count divided by the number of bytes being sampled.

The third modification is meant to address fairness problems in BLACK during periods of congestion. In BLACK, the $HitFraction$ would represent the average fraction of buffer space used by a particular flow calculated after a sample of m packets if packets were sampled from a virtual queue of size m. However, the way BLACK samples packets does not always resemble the idea of sampling from a virtual queue. BLACK samples packets triggered by packet arrivals, no matter whether the arriving packet will be dropped or enqueued. During congestion, the aggregate arrival rate might be high, and so a high level of packets drop, which is different from a serving rate. While packets are backlogged in the buffer, it is possible that the high sampling rate, as a result of a frequent packet arrival event, may cause the same packet(s) to be sampled more than once. To reduce this possible error, AFC directly collects the $HitFraction$ statistics from the packets that are enqueued and treat them in the same way as sampled packets in BLACK. After the sampling period, the $HitFraction$ of each flow could be determined using the same idea of sampling packets from the virtual queue. It is worth noticing that this new sampling method decreases the complexity of AFC in two ways. First, randomly sampling packets from the queue is no longer required because the information is directly collected from the packet that is enqueued. Second, the update frequency tends to be less because AFC collects the statistics only when there is a packet enqueued excluding those dropped packets, unlike BLACK that the statistics is collected per packet arrival.

An important aspect not considered before is the throughput fluctuation that flows experience after passing through the FAQM mechanism, in particular flows sending data from streaming applications. For example, in BLACK, whenever the $HitFraction$ is higher than the $FairFraction$, incoming packets from flow i are dropped with a certain dropping probability. This means that 1) more incoming packets from flow i might be allowed in, and 2) that already queued packets from flow i might still be in the queue. As a result, after several sampling periods, if the queue is not able to drain the old extra packets and the new packets, the $HitFraction$ may reach the maximum of twice the $FairFraction$, when the dropping probability will be equal to one. After this, the $HitFraction$ will come down to the $FairFraction$ and a new period of fluctuation will begin. AFC introduces a more aggressive dropping function and a ***credit-based mechanism*** to avoid the cyclical underutilization and overutilization of the bandwidth by unresponsive flows. In order to avoid this fluctuation, once the $HitFraction_i$ is higher than the $FairFraction$, AFC will not allow in more incoming packets from flow i, which implies a dropping probability of one. This dropping probability will continue until the extra packets are drained and the $HitFraction_i$ becomes lower than the $FairFraction$. However, dropping all the packets when a $HitFraction$ becomes higher than a $FairFraction$ might have a problem with responsive flows like TCP, which backs off when their packets are dropped. Once the $HitFraction$ of TCP traffic reaches a $FairFraction$, its incoming packets are dropped causing a back-off period to begin. After a short while, as the queue drains some packets out, the $HitFraction$ becomes lower than the $FairFraction$ once again and incoming packets are allowed to get in. However, the TCP source may still be backing off its data transmission, and thus no or only few packets would arrive at the queue causing the $HitFraction$ to be even lower. Later, after the TCP source expands its congestion window, a burst of packets once again arrives at the queue and the $HitFraction$ eventually reaches the $FairFraction$ again. In other words, the $FairFraction$ becomes an upper limit of the $HitFraction$ for TCP traffic. Since the average $HitFraction$ is less than the $FairFraction$ (See Figure 1 (left)), TCP sources cannot receive their fair share.

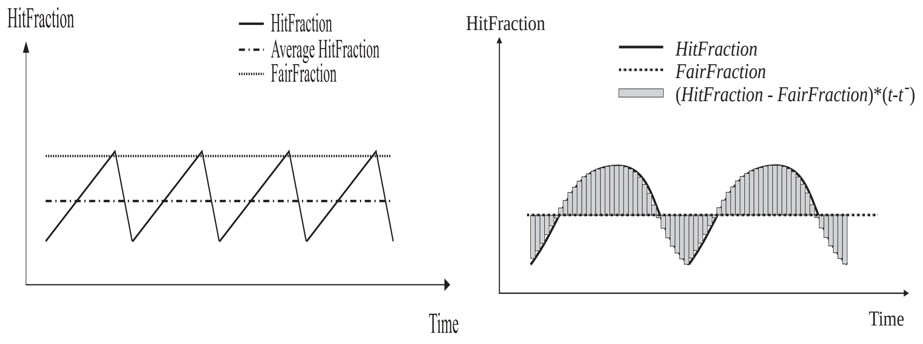


Fig. 1. The problem of aggressive dropping policy to $HitFraction$ (left) and simplified $HitFraction$ behavior of responsive traffic under new AFC dropping policy (right).

The idea of a *credit-based mechanism* in AFC is to solve this problem allowing the $HitFraction_i$ to go beyond the $FairFraction$ if flow i has credit available, so that the average $HitFraction_i$ over time is about the same as the $FairFraction$. Here, a credit is defined as the area under the $HitFraction_i$ curve above or below the $FairFraction$, which is referred to as $\Delta\mathcal{A}$. Precisely, a credit for flow i can be approximated every time a $HitFraction_i$ is updated according to

$$\Delta\mathcal{A}_t = \Delta\mathcal{A}_{t^-} + [(HitFraction_t - FairFraction_t) \times (t - t^-)] \tag{1}$$

as roughly illustrated in Figure 1, where t indicates a current update time and t^- indicates a previous update time. Obviously, the value of $\Delta\mathcal{A}_t$ should be kept as close to zero as possible. However, when a responsive flow is backing off, there is usually not enough packets enqueued to make its $HitFraction$ to raise as much as a $FairFraction$, and thus its $\Delta\mathcal{A}_t$ would become negative. The negative value of $\Delta\mathcal{A}_t$ means that this flow has this amount of credit and AFC will allow the packets of this flow to be enqueued even if its current $HitFraction$ is greater than the $FairFraction$, as long as the flow still has available credit or its $\Delta\mathcal{A}_t$ is still a negative value. This dropping policy is in contrast to the refined dropping function utilized by BLACK in which packets are dropped when a $HitFraction$ is greater than a $FairFraction$ only. By allowing a $HitFraction$ of a flow to be higher than a $FairFraction$ if it has available credit, e.g. from its previous back-off period that causes $\Delta\mathcal{A}_t$ to be negative, an underutilization of an unresponsive flow is prevented.

4 Performance Evaluation

In this section, a comparative evaluation of RED, SFB, CHOKe, BLACK, CARE, and AFC in a number of realistic scenarios is included. We take a simulation approach utilizing the ns-2 [12] simulator and the dumbbell topology shown in Figure 2. The evaluation includes scenarios with single and multiple unresponsive flows and TCP sources with different round-trip times. Because one of the design goals of AFC is to solve the problem of throughput fluctuation, not only the throughput fairness is used as a performance metric but also the instantaneous throughput of CBR traffic over time. Then, more diverse scenarios are conducted to examine fairness and robustness of the fair AQM schemes, including a scenario with traffic with different packet sizes, a scenario with short-lived and bursty traffic, and a scenario when TCP-friendly traffic and TCP traffic are sharing the same bottleneck link.

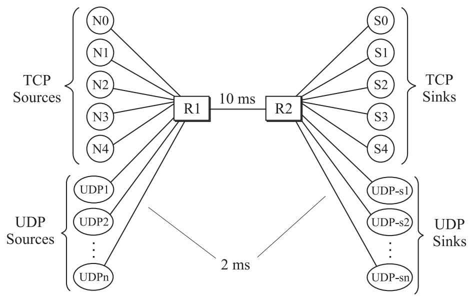


Fig. 2. Simulation topology

Each experiment is run for 200 seconds and is repeated 20 times to calculate 95% confidence intervals. However, for the sake of brevity, we do not present details of the various schemes. The statistics are collected from 50 sec. to 200 sec. The packet size is 1 Kbyte, unless explicitly stated otherwise. SFB is set with a default configuration of two levels of hash functions of 23 bins with double set of hash tables for moving hash functions (total of 46×2 bins) using the NS code provided by [13]. The min_{th} and max_{th} threshold settings for RED, CHOKe, BLACK, and AFC are 50 and 150 packets respectively which are the Gentle RED parameters [14]. In the case of CARE, the number of capture occasions (t) is 200, where 50 out of 200 can be used for the estimation of the number of flows, and the probability p_{cap} is 0.04 according to [2].

4.1 Unresponsive flows

In this section, we present simulation results using single and multiple unresponsive flows. First, a large unresponsive CBR traffic sending data at 5 Mbps shares the 5 Mbps bottleneck link with 100 TCP sources. All the access links are 100 Mbps. Then, we repeat the experiment but using 5 CBR sources. All the queue parameters are unchanged except that CHOKe is now equipped with its *self adjusting mechanism* to handle multiple unresponsive flows. With this mechanism, the region between the minimum threshold (min_{th}) and the maximum threshold (max_{th}) is divided into 8 subregions (k) and the number of packet matching in CHOKe's dropping policy performs $2 \times i$ times per each packet arrival where $i = 1..k$ is the region where the current average queue size is falling into. The Jain's fairness index [15] was utilized to calculate the fairness among the competing flows as follows:

$$f = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{2}$$

where $0 \leq f \leq 1$, and x_i is the throughput achieved by flow i. The results of the simulations are tabulated in Table 1.

Table 1. Unresponsive flows scenario

Scenario	Single unresponsive flow			Five unresponsive flows		
	Average UDP tput (Kbps)	Average TCP tput (Kbps)	Jain's Fairness index	Average UDP tput (Kbps)	Average TCP tput (Kbps)	Jain's Fairness index
RED	4,374.82	5.046	0.5551	1,000.00	0.000	N/A
CHOKe	1187.301	38.160	0.9842	841.29	5.092	0.1108
SFB, rate limit $\approx$ twice the fair rate	98.99	49.046	0.9836	95.29	45.285	0.9594
SFB, rate limit $\approx$ fair rate	49.57	49.538	0.9826	47.84	47.658	0.9611
SFB, rate limit $\approx$ half the fair rate	24.95	49.78	0.9817	22.94	48.903	0.9593
CARE	172.66	48.307	0.9862	1,000.00	0.000	N/A
BLACK	74.04	49.289	0.9871	65.02	46.378	0.9967
AFC	50.46	49.529	0.9925	48.81	47.609	0.9972

As shown in the table, both RED and CHOKe clearly show their inability to protect responsive TCP flows. TCP traffic are completely shut out in the case of RED and receive only 5.1 Kbps per connection in the case of CHOKe. CARE provides better fairness than RED and CHOKe with one unresponsive flow but its performance deteriorated considerably in the multiple flows case. The trace data from the simulation (not shown) indicates that the precision of CARE's estimation of the number of active flows is altered by the heavy load of multiple unresponsive flows. Although SFB achieves about the same level of fairness than BLACK and AFC, a network operator needs to manually set a rate limit threshold, as SFB has neither knowledge of the fair throughput nor the number of active flows. Besides, without a special per-flow treatment or a separate queue to serve the unresponsive flows, SFB cannot guarantee fairness among unresponsive flows. In the last row of the table, AFC is shown to provide much better fairness than the other schemes in both scenarios. Results not shown here also demonstrate the superiority of AFC in controlling unresponsive traffic even in the case where the arrival rate of the unresponsive flow is as much as twice the bottleneck link rate.

In addition, in terms of the throughput fluctuations for CBR traffic, this phenomenon is greatly reduced under AFC as shown in Figure 3. This is particularly important for streaming applications, which need a smooth transfer rate to perform as users expect. From the figure, it can be seen how CHOKe, BLACK and CARE have a detrimental effect on these applications. Because of the large difference in the UDP throughput over time of CHOKe, we even had to plot the results using a different vertical scale. Only SFB provides a smooth throughput comparable to AFC, while the other schemes result in highly variable throughput. Results not shown here demonstrated that AFC still provided smooth throughput to each of the five CBR sources in the second experiment.

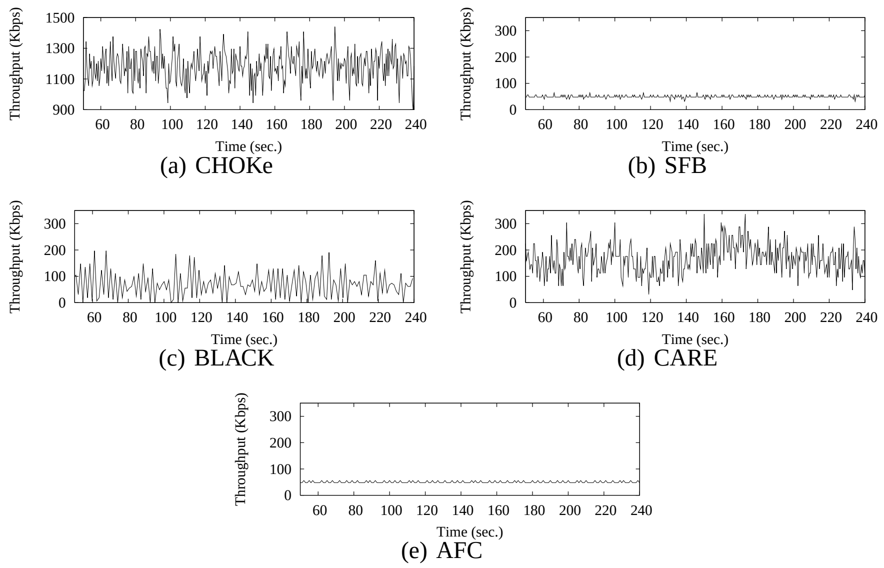


Fig. 3. CBR throughput over time after passing through different fair AQM schemes.

4.2 Traffic with different packet sizes

An experiment was set up with the same symmetric topology. One hundred TCP flows compete with three CBR flows with an arrival rate of 1 Mbps each. However, these three CBR traffic have different packet sizes, with CBR1 using a packet size of 100 bytes, CBR2 500 bytes, and CBR3 1,000 bytes. The average per-flow throughput of CBR traffic under different fair mechanisms are plotted along with a fair share of bandwidth in Figure 4. The figure clearly shows the superior fairness performance of AFC over the other schemes where the per-flow throughput of three CBR with totally different sizes of packets are provided in a fair manner.

4.3 TCP with different round-trip times

In this experiment, 200 TCP traffic are randomly originated from nodes N0, N1, N2, N3 or N4 and linked to the randomly selected sink nodes S0, S1, S2, S3 and S4 with an

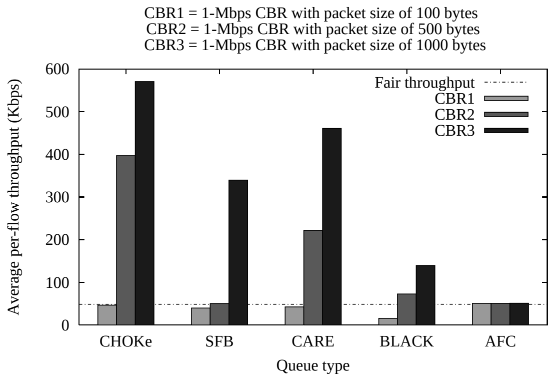


Fig. 4. Average per-flow throughput of CBR traffic with different packet sizes.

asymmetric topology where the link from N0 to R1 and the link from R2 to S0 in Figure 2 have 1 ms of propagation delay each, N1-R1 and R2-S1 5 ms, N2-R1 and R2-S2 10 ms, N3-R1 and R2-S3 15 ms, and N4-R1 and R2-S4 20 ms. The cache size of BLACK and AFC is 46 which is the same as SFB, or only a quarter of the number of these long-lived TCP traffic. The results presented in Table 2 clearly show that AFC provides the best fairness performance among TCP connections with different round-trip times.

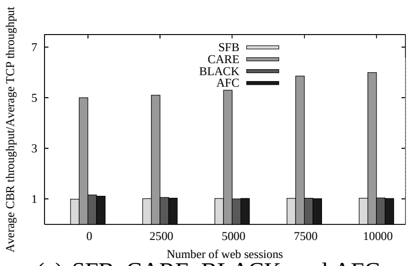
Table 2. TCP with different round-trip time scenario

	RED	CHOKe	SFB	CARE	BLACK	AFC
Jain's fairness index	0.976	0.972	0.966	0.981	0.991	0.994

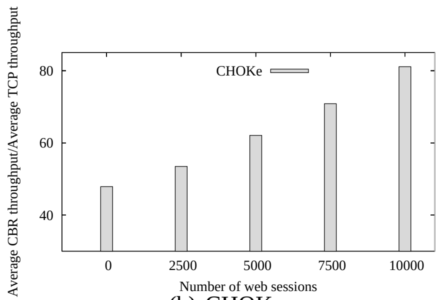
4.4 Effect of short-lived traffic

In all of the previous experiments, the fairness performance is evaluated under the ideal scenarios where all of the sources transmit unlimited amount of traffic. In this section, the schemes are evaluated using two types of short-lived traffic: 1) low-bandwidth short-lived traffic such as web traffic, and 2) high-bandwidth short-lived traffic such as malicious traffic that aims at saturating a link by escaping a fair AQM mechanism.

Low-bandwidth short-lived traffic. In this part, experiments are conducted to evaluate how fair the AQM schemes are when there are different loads of web traffic in the background. With the same topology, five CBR sources of 2.5 Mbps each and 100 TCP sources are sharing the same bottleneck link along with w sessions of web traffic. Each web session contains a default parameter recommended in the ns-2 script [12] where the traffic model is based on [16]. A Pareto distribution is used for flow lengths of web traffic where the average number of packets per flow is 15 with a shape parameter of 1.2. The starting time of each of the w web sessions is randomly set during 250 seconds of simulation time. With a fixed simulation time, a large w implies a high web traffic load scenario or more web traffic that would arrive at the queue than with a small w. In this experiments, the number of web sessions w are set to 0, 2, 500, 5,000, 7,500 and 10,000 sessions to be generated during this 250 seconds of simulation time.



(a) SFB, CARE, BLACK, and AFC



(b) CHOKe

Fig. 5. Average per-flow CBR throughput over average per-flow TCP throughput at different background web traffic loads.

Figure 5 shows the average per-flow CBR throughput over the average per-flow TCP throughput. Ideally, the ratio should be close to one to ensure fairness among different connections. The figure shows that SFB (with well-tuned rate limit), CARE, BLACK, and AFC achieve their fairness performance with minimal interference from different web traffic loads. The average CBR throughput under CARE is higher than SFB, BLACK, and AFC because of the high arrival rates that causes an underestimation of the number of active flows. However, CHOKe shows a different result. When no background web traffic is presented, the average per-flow CBR throughput is about 50 times the average long-lived TCP throughput, and the average CBR throughput gets significantly higher than the average TCP throughput as the number of web sessions increases. The rationale behind this poor performance of CHOKe is that when there are more packets from the web traffic in the queue, fewer packets from the same CBR connection are in the queue, which leads to less chance of a packet matching and less control of unresponsive traffic.

High-bandwidth short-lived traffic or bursty traffic. Although the fair AQM mechanisms achieve long-term fairness in different scenarios, no evaluation has included high-bandwidth short-lived misbehaving traffic. Fair AQM schemes that do not maintain per-flow state information might not have enough long term information about the average arrival rate of these flows and therefore might not be able to achieve fairness. Some AQM schemes need some amount of time to collect information before identifying and controlling misbehaving traffic, and may erase that information after a short while. Providing long-term fairness under high bandwidth short-lived traffic may not be possible for these lightweight fair AQM schemes, and a key question here is how well these schemes can detect and control these types of traffic.

A series of experiment are set up with misbehaving traffic represented by a high-bandwidth short-lived UDP traffic with different ON and OFF periods, which are drawn from an exponential distribution. Two sets of experiments are set for two peak rates of

1 Mbps and 10 Mbps. Again, 100 long-lived TCP sources are passing through the same bottleneck link of 10 Mbps. The results are summarized in Table 3.

Table 3. Scenario with 1 Mbps and 10 Mbps peak rate bursty traffic.

	1 Mbps peak rate bursty traffic scenario				10 Mbps peak rate bursty traffic scenario			
No. of UDP	5 flows		15 flows		5 flows		15 flows	
	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)	Avg. UDP tput (Kbps)	Avg. TCP tput (Kbps)
RED	456.9	77.2	405.0	39.4	1698.9	9.8	666.3	0.0
CHOKe	385.2	80.8	364.4	45.5	893.9	55.3	653.3	1.9
SFB	321.1	84.0	375.6	43.8	86.1	95.7	115.6	82.7
BLACK	257.1	87.2	264.2	60.5	595.7	68.9	385.7	37.2
CARE	213.0	89.4	264.8	60.4	1098.7	44.4	666.4	0.0
AFC	265.7	86.7	245.1	63.4	231.4	87.7	191.1	68.4

For bursty traffic with 1 Mbps peak rate case and 5 CBR flows, although all of the schemes leave the bandwidth to each bursty traffic at least 1.8 - 3 times the fair share, all of the schemes still provide some level of protection to TCP connections. However, with 15 CBR bursty sources, RED clearly provides the least fairness, as all bursty traffic altogether take up to 60% of the total bandwidth and leave each TCP connection with a bandwidth equal to only about half of the fair share. Here, CARE estimates the number of active flows much better as the arrival rate of the bursty traffic is smaller. The result of SFB is different because SFB needs to know a rate limiting threshold in advance, which should be manually configured. Using the same settings that achieve good fairness in the 10 Mbps peak rate case, SFB turns to provide poorer fairness performance in this 1 Mbps peak rate case. This result shows that although SFB could detect unresponsive traffic, it cannot use the same settings to provide fairness for different scenarios.

For bursty traffic with 10-Mbps peak rate, the RED mechanism cannot protect TCP traffic in all the scenarios. CHOKe and CARE could provide some protection when there are 5 bursty flows, but cannot prevent 15 bursty flows from grasping almost all of the bandwidth. It is expected for CHOKe as the results in the previous experiments show that CHOKe does worse in providing fairness under a higher number of unresponsive flows. CARE, however, is a little worse than CHOKe because of its inability to estimate the number of active flows correctly with unresponsive traffic with high arrival rates. BLACK provides some level of protection as it can reserve about 60% - 70% of the fair throughput to each TCP connection on average when there are 5 bursty traffic. A trace file indicates that BLACK's $HitFraction$ mechanism, that randomly samples packets from the queue, does not perform as well as when the unresponsive traffic is non-bursty. On the other hand, the direct counting of $HitFraction$ of AFC provides better fairness, as each TCP connection gains more than 92% of the fair share. However, both BLACK and AFC performance are degraded when there are 15 bursty traffic coming to the queue because of the cache size of only 20. In this case, bursty traffic does have an impact on the mechanism of BLACK and AFC, as that traffic could be replaced easily during the OFF period of the traffic. Here, BLACK turns to be largely distorted, while AFC is still far better as each TCP connection gains 72% - 90% of the fair share. On the other hand, SFB performs well in these scenarios, but with a fine tuning of rate limiting given that the arrival rate of bursty traffic is known in advance.

5 Conclusion

This paper proposes AFC, a novel fair AQM scheme that outperforms current schemes in well-known as well as new and more realistic scenarios not utilized before. Using simulations, we evaluate and compare AFC with RED, SFB, CARE and BLACK including not only a single unresponsive flow but also multiple unresponsive flows, TCP connections with different RTTs, low and high bandwidth bursty background traffic, and different packet sizes. We show that AFC provides the best fairness performance in all these cases with low complexity and memory requirements. In addition, AFC's credit-based mechanism avoids oscillations and underutilization of responsive flows and provides smooth transfer rates to unresponsive flows.

References

1. S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Trans. Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999.
2. M. Chan and M. Hamdi, "An Active Queue Management Scheme Based on a Capture-Recaptured Model," *IEEE J. Select. Areas Commun.*, vol. 21, no. 4, May 2003.
3. H. T. Kung, T. Blackwell, and A. Chapman, "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing," in *Proc. ACM SIGCOMM*, Aug. 1994, pp. 101–114.
4. R. Pan, B. Prabhakar, and K. Psounis, "CHOKe - A Stateless Active Queue Management Scheme For Approximating Fair Bandwidth Allocation," in *Proc. IEEE INFOCOM*, April 2000, pp. 942–951.
5. W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness," in *Proc. IEEE INFOCOM*, April 2001, pp. 1520–1529.
6. G. Chatranon, M. A. Labrador, and S. Banerjee, "BLACK: Detection and Preferential Dropping of High Bandwidth Unresponsvie Flows," in *Proc. IEEE ICC*, May 2003, pp. 664–668.
7. S. Floyd and V. Jacobson, "Random Early Detection Gatewas for Congestion Avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
8. K. Psounis, R. Pan, and B. Prabhakar, "Approximate fair dropping for variable-length packets," *IEEE Micro*, vol. 21, no. 1, January/February 2001.
9. G. Chatranon, M. A. Labrador, and S. Banerjee, "A Survey of TCP-Friendly Router-based AQM Schemes," *Computer Communications*, vol. 27, no. 15, pp. 1424–1440, August 2004.
10. I. Kim, "Analyzing Network Traces To Identify Long-Term High Rate Flows," Master's thesis, Texas A&M Univ., May 2001.
11. C. Estan, G. Varghese, and M. Fisk, "Counting the Number of Active Flows on a High Speed Link," *ACM SIGCOMM Computer Communication Review*, vol. 32, July 2002.
12. "NS Network Simulator," http://www.isi.edu/nsnam/ns/.
13. "The Network Simulator: Contributed Code," http://www.isi.edu/nsnam/ns/ns-contributed.html.
14. S. Floyd, "Recommendation on using the "gentle_" variant of RED," 1999, http://www.icir.org/floyd/red/gentle.html.
15. R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.
16. A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proc. ACM SIGCOMM*, Sept. 1999.