# Performance Evaluation of Fairness-Oriented Active Queue Management Schemes

Meriel Huggard, Mathieu Robin, Arkaitz Bitorika, Ciarán Mc Goldrick
Department of Computer Science
Trinity College Dublin, Ireland
{huggardm, robinm, bitorika, cmcgldrk}@cs.tcd.ie

## Abstract

*Active Queue Management (AQM) schemes are a class of queueing algorithms designed to surmount some of the shortcomings of classic Drop-Tail queues in best-effort networks. Most AQM algorithms are primarily designed to improve congestion control through early notification. However significant work has been done on more advanced AQM schemes designed to protect responsive flows against unresponsive traffic. Such schemes are designed to identify and penalise unresponsive flows, which may use an unfair share of the available resources.*

*In this study, six algorithms are evaluated through simulation-based experiments. The schemes chosen employ lightweight mechanisms for approximating fair bandwidth sharing. They are designed to be scalable and allow for incremental deployment in the current best-effort Internet infrastructure. Evaluation of their performance is effected under various network traffic conditions. The operational complexity of the schemes is also assessed.*

## 1. Introduction

Active Queue Management denotes the class of algorithms designed to provide improved queueing mechanisms for network routers. Research in this area grew out of the original RED proposal [13]. These schemes are called *active* because they dynamically signal congestion to sources; either explicitly, by marking packets (e.g. Explicit Congestion Notification [11]) or implicitly, by dropping packets. The Internet Engineering Task Force (IETF) recommended the deployment of AQM in Internet routers in RFC 2309 [4]. The main motivations were cited as the improvement of performance due to smaller overall queueing delays, reduced packet drop rates and prevention of synchronisation due to the drop-tail queueing process [4]. RFC 2309 also recommended the development of new queueing mechanisms to improve fairness for best-effort Internet traffic, and this gave rise to a large body of work targeted at creating new AQM proposals.

While RED and many of the other proposed AQM schemes address these issues; they do not prevent the problem of congestion collapse that may arise from the growth of non-responsive traffic on the Internet [4, 12]. Flows that do not use TCP-compatible end-to-end congestion control could pose a threat to Internet performance because they are able to obtain an unfair share of the network bandwidth.

There is ongoing research into congestion-control protocols for applications that do not require the reliability provided by TCP, such as streaming media delivery platforms. Some have argued that TCP could be used for such a purpose [19], while others have suggested that multimedia streaming is fair to all other data flows [6]. In general, these applications have not used any congestion control or have incorporated their own approach, such as RTP [26]. Much of the research has been targeted at designing so-called "TCP-Friendly" protocols to transport datagrams in a unreliable but responsive way. Such protocols include GAIMD, TFRC, and TEAR [28]. DCCP [18] is a more recent transport protocol designed to provide an unreliable flow of datagrams in tandem with a choice of TCP-friendly congestion control mechanisms. Overall, these mechanisms have yet to achieve wide scale deployment. Unfortunately, given the nature of the Internet, there is no way to enforce good behaviour among unresponsive flows.

A subset of existing AQM proposals have been designed to provide better fairness and protection for flows that use end-to-end congestion control, usually by limiting the rate of unresponsive flows. In this paper, six schemes that employ fairness-oriented metrics, are evaluated using a specially developed framework which uses the NS2 simulator [2]. RED is also included

Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)
1526-7539/04 $20.00 © 2004 IEEE

in this study for baseline comparison purposes. The experimental evaluation utilises a known set of traffic conditions, where the fairness and ability to protect responsive flows is evaluated for each algorithm. The selected algorithms are designed to be lightweight, incorporating mechanisms to avoid costly per-flow tracking. Nonetheless, performance remains an issue and hence their enqueueing complexity is also considered as part of this study.

## 2. Scope of the Study

The six algorithms evaluated in this study have been selected according to the following criteria:

- The schemes can be deployed directly, and incrementally, in place of current router implementations, without the need to modify Internet protocols on the end nodes.
- The schemes use no per flow tracking, to allow for efficient use with high-throughput backbone routers.
- The algorithms must be sufficiently documented in the peer-reviewed literature to allow consistent implementation.
- Only schemes targeted at best-effort IP networks are considered.

The algorithms selected for this study are described below:

**RED** [13] is not specifically fairness-oriented. RED has been included in this study as it is the AQM recommended by the IETF for deployment [4]. The RED algorithm estimates the current congestion level on the network by keeping track of the average queue size. This average is computed using an exponential weighted moving average (EWMA), which acts as a first order low pass filter [23], so short bursts of traffic do not affect the way RED drops packets. Sources are notified of congestion, typically through packet dropping or marking. The probability of notification is an increasing function of the average queue length, as shown in figure 1.

**CHOKe** [24] augments RED by providing a stateless mechanism for approximating fair bandwidth allocation of the traffic traversing a link. CHOKe extends the RED probabilistic dropping algorithm: an arriving packet is compared with a random packet drawn from the queue and they are both dropped if they belong to the same flow. If they are from different flows, the arriving packet is subjected to the standard RED drop probability. CHOKe is more effective against unresponsive flows when more than one candidate packet is compared for each packet arrival. The number of comparisons made can be adjusted depending on the congestion level.

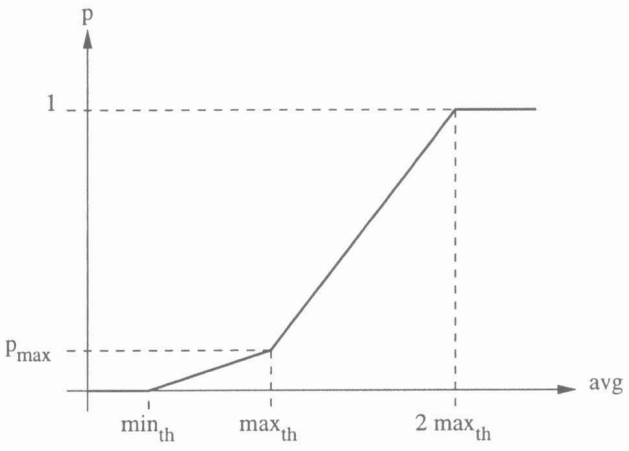



**Figure 1. Drop probability of RED as a function of the estimated queue length**

**RED-PD** [21] is a variant of RED that uses the RED packet drop history to identify high-bandwidth flows. Considering the packet drop history as a random sample of the traffic at the link, it keeps $M$ lists with recent drop histories. Flows are identified as being aggressive when they appear in at least $K$ of the $M$ drop history lists. These flows are then monitored and a flow-specific dropping probability is applied before the RED drop decision is taken. This pre-filter provides relative fairness among the monitored flows and limits the bandwidth obtained by them.

**LRU-RED** [25] extends RED by keeping a cache of recently seen flows that uses the Least-Recently-Used (LRU) replacement policy. High-bandwidth flows are identified because they are more likely to remain in the cache. If the value associated with a flow's entry in the cache exceeds a given threshold, its RED drop probability is increased with the goal of penalising high bandwidth flows.

**Stochastic Fair Blue (SFB)** [10] is an extended and improved version of the BLUE [8] algorithm. While BLUE dynamically adjusts a global drop probability depending on the queue occupancy, SFB applies a similar mechanism to adapt the drop probability differently for every flow. Using a Bloom filter [3], it can efficiently map the

different flows traversing the buffer to their respective drop probabilities.

**Self Adjustable CHOKe (SAC)** [17] is an extension of CHOKe that intends to improve on CHOKe in two regards. Firstly, it automatically adjusts the number of random samples taken from the queue for each arriving packet. Secondly, when the compared flow IDs are equal it does not drop the random sample. Instead, a drop candidate from the same flow is selected by searching beginning from the tail of the queue, with the goal of providing better fairness between unresponsive flows.

**CARE** [5] provides an algorithm based on a capture-recapture model to approximate a fair distribution of the available bandwidth. It randomly "captures" packets and, based on this sample, estimates the number of flows and their respective bit rate. Using this information, the drop probability for every flow is computed as an increasing function of the estimated share of the resources used by a given flow.

## 3. Methodology

### 3.1. Simulation Methodology

The six algorithms presented in the previous section were evaluated using the simulation framework described in [2]. This framework has been developed to provide for the evaluation of AQM schemes through a powerful user interface to NS2. Its Python API allows the user to set up a large number of NS2 simulations, and to aggregate, analyse and summarise the results. The framework can improve the statistical significance of the results by the appropriate processing of output data [2]. In doing so, it removes the initial transient using the Marginal Confidence Rule described in [27], and determines the simulation run-length dynamically. To achieve a given level of confidence, it also computes the number of replications for each simulation dynamically by employing the *replication/deletion approach* [20]. The results presented in this paper were obtained using a relative confidence interval of, at most, $\pm 15\%$ of the mean, with 95% confidence. The results included in this work were obtained using the latest available version, 2.27, of the NS2 simulator.

### 3.2. Network Scenario

A network scenario is defined as the combination of a traffic mix and a network topology. The traffic is a mix of long-lived TCP flows (e.g. FTP transfers of large files) and UDP flows to simulate multimedia traffic. The long-lived TCP flows and UDP flows are active at all times during the simulation. The network topology comprises a single, congested 10Mbps link (see Figure 2 ). The Round-Trip Times are uniformly distributed between 15ms and 500ms. The bounds of this distribution are taken from Internet measurements [1].

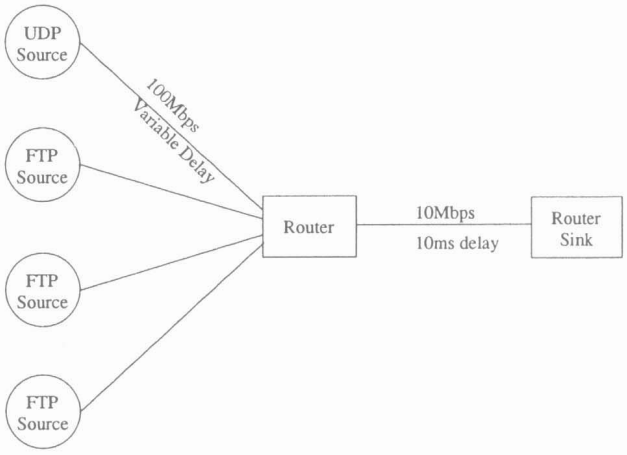



**Figure 2. Dumbbell Topology**

### 3.3. Parameter settings

One of the major difficulties encountered was in ensuring the consistency and the fairness of this evaluation. The AQM schemes can be tuned by adjusting some internal parameters, and each one influences the algorithm's performance. By way of example, a recent study [7] has shown that the choice of the queue unit for RED (*bytes* or *packets*) can have a significant impact on its performance. Much criticism of the original RED proposal stemmed from the parameterisation guidelines provided, as these were not always appropriate for a wide range of network scenarios [22, 9, 14].

In the schemes under consideration, most authors provide parameterisation guidelines. However, these are not always complete and, in some cases, must be inferred from the evaluation scenarios described in the respective AQM proposal. Some guidelines assume *a priori* knowledge of the traffic model and characteristics (e.g. mean packet size for RED in byte mode). Routers that use these algorithms may experience problems when faced with unpredictable Internet traffic that fails to conform to the adopted model.

The notation used is that of the original authors. The AQM parameters are set according to their authors' recommendations, see Table 1.

Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)
1526-7539/04 $20.00 © 2004 **IEEE**

| AQM | Parameters |
|---|---|
| RED | $w_q = 0.004$, $min_{th} = 5$, $max_{th} = 15$ |
| CHOKe | Same parameters as RED. The number of regions, used to determine the number of drop candidates, is set to 2. |
| CARE | The size $t$ of the capture list is set to 200, and the probability of capture $p_{cap}$ to 0.04. |
| RED-PD | Same parameters as RED, with RED-PD specific parameters set to: $K = 3$, $M = 5$, target RTT $R = 250$ ms. |
| LRU-RED | Same parameters as RED. The cache size is set to 30, probability to $\frac{1}{40}$, $\Delta$ to 7.5ms, and the threshold to 135. |
| SAC | Same parameters as CHOKe, with SAC-specific parameters set as follows: the weighting parameters $w_k$ and $w_p$ are set to 0.005 and 0.004, and the initial number of regions $k$ is set to 10. |
| SFB | $\Delta = 0.0005$, $L = 2$, $N = 23$. The hash function for level $i \in [1, L]$ of the Bloom filter is defined by: $h_i(k) = \lfloor N\,(r_i k \pmod 1)) \rfloor$, where $k$ is the flow identifier, and $(r_i)$ a sequence of random numbers uniformly distributed over $[0, 1]$. |

**Table 1. AQM Algorithm Parameterisation**

## 4. Performance Evaluation

The algorithms are evaluated for different network scenarios. The two main metrics of interest are fairness and utilisation. In this study, Jain's fairness index [16] is applied to throughput of individual flows and is the main fairness metric used. The fairness $\phi$ is defined as follow: if there are $N$ flows competing for the bandwidth, and each flow is characterised by its average throughput $s_i, i \in [1, N]$, then:

$$\phi = \frac{\left(\sum_{i=1}^{N} s_i\right)^2}{N \sum_{i=1}^{N} s_i^2},$$

Utilisation is also relevant in this evaluation: the results presented show that many of the schemes, by focusing on fairness, give rise to situations where the outgoing link is greatly underutilised. The utilisation metric is defined as the percentage of total available network capacity utilised in one direction during a simulation run.

### 4.1. Single unresponsive flow

The effectiveness of the AQM schemes at protecting TCP traffic from non-responsive UDP flows is evaluated in an environment where constant-bitrate UDP traffic is introduced to compete with existing FTP flows. The network scenario is characterised by a bottleneck bandwidth of 10Mb/s, and background traffic arising from 50 long-lived FTP flows. The performance of the AQM schemes is evaluated by testing different configurations, where the rate of a single UDP flow varies from 0 to 16Mb/s. It is expected that the fairness will decrease, as the unresponsive UDP flow will consume as much bandwidth as it needs, while the 50 responsive TCP flows will share the remaining bandwidth.

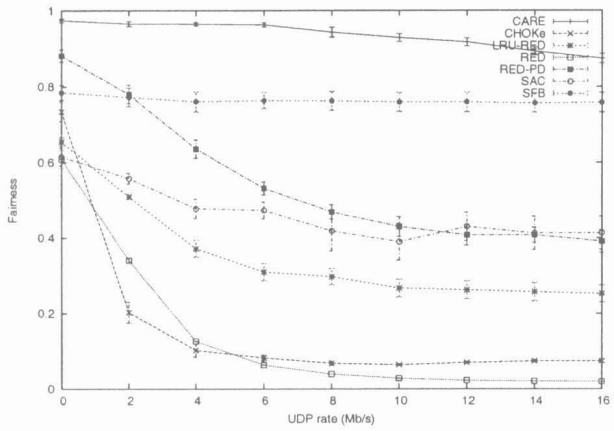
Figure 3 shows the effects of the increasing load of UDP traffic on two performance metrics, fairness and utilisation. As expected, the fairness decreases as the offered unresponsive traffic increases. However, the performance of the different algorithms is diverse and a clear hierarchy appears:

- CARE provides very good fairness, approximately in the range [0.9 - 1]. While it is affected by the introduction of unresponsive traffic, its performance appears much better than that of the other schemes.
- SFB maintains almost constant fairness in the range [0.7 - 0.8], and is not significantly affected by the change in the traffic mix.
- For RED-PD, SAC, and LRU-RED the fairness is affected by the UDP load, decreasing gradually as the UDP rate increases.
- Finally, CHOKe essentially fails to penalise the unresponsive flow, offering only a marginal gain over RED. RED does not include any mechanism to improve fairness.

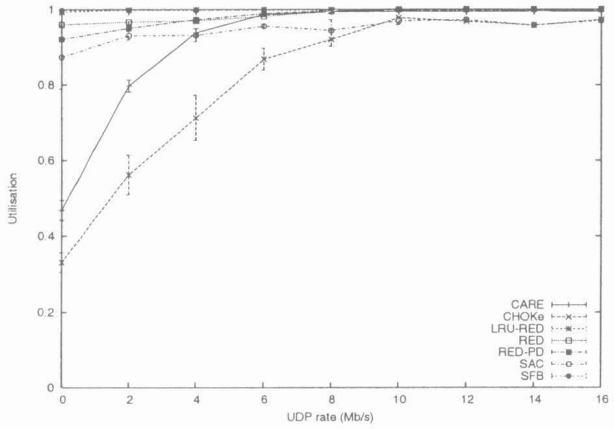
In improving the fairness, some schemes are too aggressive in dropping packets, thus leading to non-optimal utilisation of the bottleneck link. Figure 3(b) shows that CARE, which is able to achieve a very good approximation of a fair distribution of the bandwidth, provides for increasing utilisation as the UDP bitrate increases. This finding applies to CHOKe, and, to a lesser extent, to SAC.

### 4.2. Multiple unresponsive flows

In this scenario, the single unresponsive flow is replaced by an increasing number (from 0 to 16) of UDP flows, each transmitting datagrams at the rate of 1Mb/s. While the level of unresponsive traffic is the




(a) Fairness



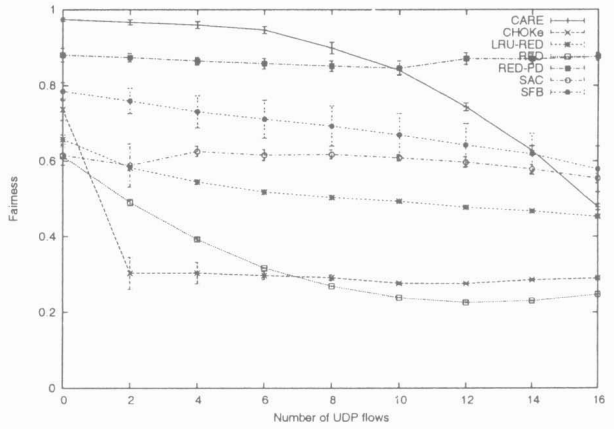

(b) Utilisation

**Figure 3. Fairness and utilisation measurements with 50 FTP flows and a single constant-bitrate UDP flow**

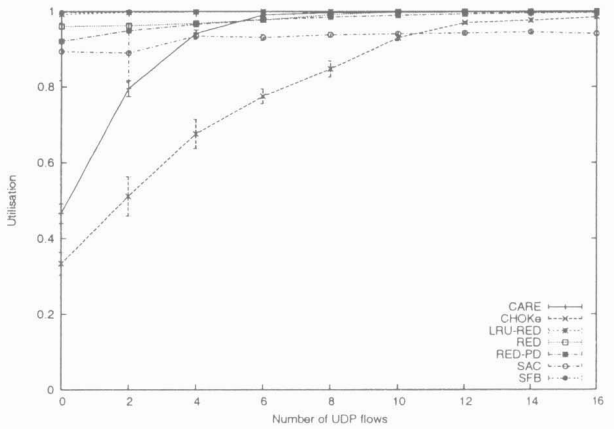
same as in the previous experiment, it becomes harder to detect. Each unresponsive flow is exceeding its fair share of the bandwidth by a smaller margin. The results are shown in Figure 4. Utilisation is also very similar to that in the single-UDP flow case. For fairness see (Figure 4(a)), the hierarchy is slightly changed:

- CARE obtains good fairness when the UDP load is low, but its performance deteriorates when new unresponsive flows are introduced because it underestimates the number of flows in the scenario.
- SFB and RED-PD achieve the most consistent fairness. RED-PD performs much better than in the previous scenario, while SFB's performance degrades slowly as new unresponsive flows are introduced.
- LRU-RED and SAC are mostly successful at keeping a stable fairness.
- CHOKe's performance is generally poorer than tat of RED, while also having one of the overall lower utilisations.

RED-PD offers the best performance in this scenario. It manges to keep the utilisation above 90% in all cases, while consistently providing for a good fairness. On the other hand, SFB still provides for near 100% link utilisation, even if its fairness is lower.




(a) Fairness



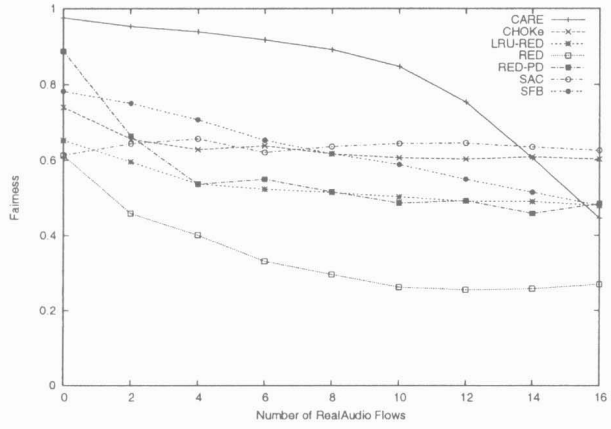

(b) Utilisation

**Figure 4. Fairness and utilisation metrics with 50 FTP flows and increasing number of constant-bitrate 1 Mb/s UDP flows**

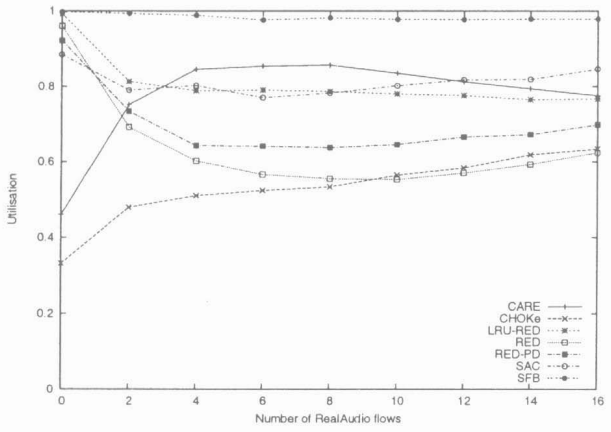

1526-7539/04 $20.00 © 2004 **IEEE**

### 4.3. RealAudio Traffic

In this experiment, the simple constant-bitrate UDP traffic is replaced by a more complex multimedia stream model. The methodology remains the same: different traffic mixes are used for the evaluation, with an increasing unresponsive traffic load. This traffic is generated using the NS2 built-in RealAudio traffic generator. This traffic format is basically a random ON-OFF pattern, with ON time based on an empirical distribution. UDP is used as the transport protocol. In this experiment, a RealAudio flow sends data at a rate of 1Mb/s during ON times.




(a) Fairness




(b) Utilisation

**Figure 5. Fairness and utilisation metrics with 50 FTP flows and increasing number of RealAudio flows (for clarity, error bars have been omitted).**

The results are shown in Figure 5. CARE stands out as providing the best fairness when the number of unresponsive flows is low, while SFB offers by far the best link utilisation at the expense of lower fairness. RED-PD is also not able to achieve the good fairness that it obtained in the previous experiment. For all the other schemes, the utilisation is much lower than in the previous experiments.

## 5. Operational Complexity

One of the considerations needed when deploying an AQM scheme is its potential impact on the throughput of the router. The increased queueing logic can slow down routers that are primarily designed to use basic drop-tail queueing algorithms. The notion of cache is different for each algorithm. In general, caching provides a mechanism to identify flows using an unfair share of the available resources, without the use of explicit, and costly, per-flow tracking.

### 5.1. Scalability

The complexity of the enqueueing operation for the selected schemes is now considered. This is expressed as a function of the given parameters (see Table 1), and presented in Table 2.

- CHOKe requires at most $2n$ comparisons per queueing operation, where $n$ is the number of regions.
- LRU-RED require a fixed number of operations per queue update: the cache is indexed through a hash-table.
- For SAC the complexity is dependent on the number of congestion contributing unresponsive flows. These flows increase both the number of drop candidates and the resultant linear searches of the queue that SAC performs.
- For RED-PD, the number of operations per enqueue is roughly proportional to the drop history list size.
- For each enqueue, CARE counts the number of occurrences of the arriving packet's flow in the capture list. Thus the complexity is roughly linear with the size, $t$, of this list. Additionally, CARE has to compute an estimate of the number of flows each time the capture list is updated. This operation is also linear with $t$.
- Finally, SFB maintains a Bloom filter [3], which is updated during each queueing operation. Thus the complexity is linear in the number of level $L$ of the filter.


1526-7539/04 $20.00 © 2004 **IEEE**

| AQM | Complexity |
|---|---|
| LRU-RED | Fixed number of operations. |
| CHOKe | $O(n)$, where $n$ is the number of comparisons made for each packet arrival. |
| CARE | $O(t)$, where $t$ is the size of the capture list. |
| RED-PD | $O(K)$, the length of the drop history list is proportional to $K$. |
| SAC | $O(N)$, where $N$ is the number of congestion contributing flows. |
| SFB | $O(L)$, where $L$ is the number of levels of the Bloom filter. |

**Table 2. AQM Algorithms complexity**

### 5.2. Memory requirements

The schemes studied here typically need to maintain state information in addition to variables such as drop probability or queue occupancy averages used in typical AQM schemes. These result in important memory overhead in the router.

- CHOKe and SAC are stateless because they use the queue itself as a representation of the flows traversing the router.
- LRU-RED uses a cache of fixed maximum size containing flow information with entries composed of a flow ID, a packet count and a time stamp. The memory footprint of LRU-RED is thus proportional to the cache size, which is set to 30 in the experiments detailed above.
- CARE keeps a fixed-sized cached which stores flow IDs of recently captured flows. CARE's parameter $t$ gives the size of the cache, which is configured as $t = 200$ in this work.
- RED-PD keeps a recent drop history list. The period covered by this list is proportional to $K \times R$. Thus the actual memory usage will depend on $K$, $R$, and the arrival rate at the queue.
- The Bloom filter associated with SFB keeps a fixed-size state of $N \times L$ bins of integers. Using values of 23 and 2 respectively for N and L, this yields a state of size 46 integers for the experiments described herein.

For all schemes, the memory usage is bounded and explicitly configurable through the associated algorithm parameters (see Table 1).

### 5.3. Algorithm profiling

The schemes are profiled using gprof [15], through their implementations in NS2 . This approach provides the average time each scheme spends queueing packets, as this is the operation at which the AQM queueing logic is added.

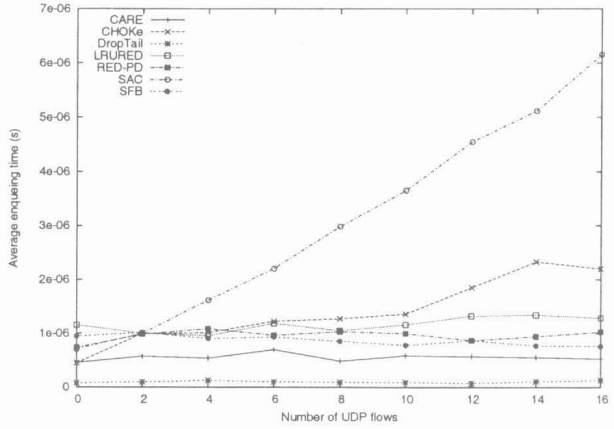



**Figure 6. Average time per enqueueing operation for AQM Schemes with multiple UDP flows**

Figure 6 show the profiler results, for a scenario where the number of unresponsive UDP flows is increasing. It can be seen that all AQM schemes incur a considerable overhead when compared to drop-tail queueing. CARE is consistently the most efficient AQM. The queueing time associated with CHOKe increases slightly with the introduction of high-bandwidth unresponsive flows. SAC shows an approximately linear increase, consistent with the scalability assessment in Table 2. The remaining schemes show stable profiler times, bounded by those of CARE and CHOKe. With the exception of SAC, all the schemes appear to be scalable. Optimised hardware-based implementations are likely to improve the performance of the fairness-oriented AQM schemes.

## 6. Conclusion

The performance of six AQM schemes designed to provide better fairness and to protect responsive flows has been evaluated. While all improve upon the fairness provided by RED, CARE offers the best approximation of fair bandwidth sharing when the number of

UDP flows is low. However, it is not as efficient when there are multiple constant-bitrate UDP flows. Overall, SFB stands out as being the algorithm which provides consistently good fairness/utilisation trade-off across the different network scenarios described herein.

Even though the complexity of the fairness-oriented AQM schemes is notably higher than Drop-Tail queueing, with the exception of SAC there are no significant differences in scalability among the schemes. The algorithms employed are in general scalable, both in terms of computational complexity and memory usage. Thus, a widespread deployment of schemes such as SFB and CARE in core routers is both possible and desirable.

## 7. Acknowledgements

This work has been supported by Enterprise Ireland Basic Research Grant, SC/2002/293.

## References

[1] M. Allman. A web server's view of the transport layer. *ACM Computer Communication Review*, 30(5):10–20, Oct. 2000.

[2] A. Bitorika, M. Robin, and M. Huggard. An evaluation framework for active queue management schemes. In *Proc. MASCOTS'03*, pages 200–206. IEEE, Oct. 2003.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[4] B. Braden et al. Recommendations on queue management and congestion avoidance in the Internet. RFC 2309, Apr. 1998.

[5] M.-K. Chan and M. Hamdi. An active queue management scheme based on a capture-recapture model. 21(4):572–583, May 2003.

[6] R. Doshi and P. Cao. Streaming traffic fairness over low bandwidth WAN links. In *Proceedings of WIAPP*, pages 30–34. IEEE, June 2003.

[7] W. M. Eddy and M. Allman. A comparison of RED's byte and packet modes. *Computer Networks*, 42(3):261–280, June 2003.

[8] W. Feng, D. Kandlur, D. Saha, and K. G. Shin. BLUE: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, University of Michigan, Apr. 1999.

[9] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. A self-configuring RED gateway. In *Proc. INFOCOM'99*, volume 3, pages 1320–1328. IEEE, Mar. 1999.

[10] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. Stochastic Fair Blue: A queue management algorithm for enforcing fairness. In *Proc. INFOCOM'01*, volume 3, pages 1520–1529. IEEE, Apr. 2001.

[11] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5):10–23, 1994.

[12] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. 7(4):458–472, Aug. 1999.

[13] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. 1(4):397–413, Aug. 1993.

[14] S. Floyd and E. Kohler. Internet research needs better models. *ACM Computer Communication Review*, 33(1):29–34, Jan. 2003.

[15] S. Graham, P. Kessler, and M. McKusick. gprof: A call graph execution profiler. In *Proceedings of the SIGPLAN '82 Symposium on Compiler Construction*, volume 17 of *6*, pages 120–126, June 1982.

[16] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report DEC-TR-301, Digital Equipment Corporation, Sept. 1984.

[17] Y. Jiang, M. Hamdi, and J. Liu. Self adjustable CHOKe: An active queue management algorithm for congestion control and fair bandwidth allocation. In *Proc. ISCC'03*, pages 1018–1025. IEEE, July 2003.

[18] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion control without reliability. Technical report, ICSI Center for Internet Research, 2003.

[19] C. Krasic, K. Li, and J. Walpole. The case for streaming multimedia with TCP. In *8th International Workshop on Interactive Distributed Multimedia Systems (iDMS 2001)*, Sept. 2001.

[20] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw Hill, third edition, 2000.

[21] R. Mahajan and S. Floyd. Controlling high bandwidth flows at the congested router. Technical Report TR-01-001, AT&T Center for Internet Research at ICSI (ACIRI), Apr. 2001.

[22] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. In *Proc. IWQoS'99*, pages 260–262, June 1999.

[23] V. Misra, W. B. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of ACM/SIGCOMM*, pages 151–160. ACM, Aug. 2000.

[24] R. Pan, B. Prabhakar, and K. Psounis. CHOKe — a stateless active queue management scheme for approximating fair bandwidth allocation. In *Proc. INFOCOM'00*, pages 942–951, Mar. 2000.

[25] S. A. L. N. Reddy. LRU-RED: an active queue management scheme to contain high bandwidth flows at congested routers. In *Proc. GLOBECOM'01*, volume 4, pages 2311–2315. IEEE, Nov. 2001.

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 3550, July 2003.

[27] K. P. White. An effective truncation heuristic for bias reduction in simulation output. *Simulation*, 69(6):323–334, Dec. 1997.

[28] Y. Yang, N. S. Kim, and S. Lam. Transient behaviors of TCP-friendly congestion control protocols. In *Proc. INFOCOM'01*, volume 3, pages 1716–1725. IEEE, Apr. 2001.