On the design of AQM's to achieve asymptotic fairness between competing TCP flows

Rade Stanojević and Robert Shorten

June 6, 2005

Abstract

Providing fairness among users with different levels of aggressiveness and different levels of responsiveness is necessary for efficient network flow control. Current solutions require keeping explicit per-flow states, multiple queues, or a change in IP architecture. In this paper we propose two new First Come First Serviced approaches for solving this fairness problem without keeping explicit per flow state. In the first approach, a spectrum of stateless AQM schemes that are similar in spirit to CHOKe are presented. We refer to these AQM's as MLC (multi-level comparison) AQM's. We show that using MLC, bandwidth allocation amongst TCP users in the network can be made arbitrarily close to max-min fairness by adjusting the MLC parameters. Our second strategy, called MAY (Markov Active Yield), is an AQM that keeps a history of packets that have been dropped from the queue and uses this list to determine which future packets should be dropped. Surprisingly, we show that this approach leads to max-min fair bandwidth allocation among an arbitrary number of AIMD flows. The properties of both MLC and MAY are evaluated and characterized theoretically using a Markov chain model, and experimentally using packet level ns-2 simulations.

1 Introduction

Active queue management (AQM) has been a topic of research in the networking community for over a decade. The potential benefits of AQM are compelling: (i) efficient utilization of network resources; (ii) friendliness to short-lived flows; (iii) the potential to accommodate bursty network traffic without excessive network delays; (iv) the ability to implement QoS strategies; and (v) the ability to police networks and provide fairness among users. Despite these, and over a decade of active research on this topic, AQM based routers have yet to be widely deployed in the internet. More worrying, there also does not yet appear to be a general consensus in the community as to whether AQM brings any real benefits to communication networks. The inability of the research community to reach a consensus appears, at least in part, to be due to two reasons. Firstly, it seems to be the case, that while AQM's that achieve some of the above network properties are relatively straightforward to realize, realizing AQM's to achieve other network properties is extremely difficult. For example, stabilizing the average queue length (and hence network jitter and latency) in RED has to-date proved to be an elusive goal in AQM research. It appears that lack of progress with some of these more difficult problems has been perceived in an overly negative manner by the community and has caused some of the other more easily realisably benefits of AQM to be overlooked. A second problem appears to be a general lack of understanding as to how many AQM's actually work in terms of throughput allocation among users. Despite a decade of research, little supporting theory exists to support their design, and their construction, while often ingenious, are more often *ad-hoc* than principled.

One of our objectives in this paper is to address the second of these issues. We are mainly concerned with characterizing the throughput and fairness properties of networks of TCP flows that compete for bandwidth via AQM based routers with certain characteristics. By developing, and analyzing Markov chain models, that describe certain classes of AQM's, we partially achieve this objective and obtain formulae that accurately characterize the asymptotic throughput of TCP users in networks employing certain AQM's. We also show that our results can be used to support the design of new AQM's with pre-defined levels of network fairness, and demonstrate how these can be implemented without using explicit per-flow state information.

1.1 Background and main contributions

Allocating bandwidth fairly amongst competing users in the Internet is important task. The central point in the current Internet is the fact that most of traffic is generated by responsive TCP users. However, the level of responsiveness of an individual user might depend on implementation of transport protocol as well as global characteristics of flow such as round trip time or queueing delays. Schemes to ensure fair bandwidth allocation would play a role of protecting less aggressive users from aggressive ones and therefore would allow existence of various end-to-end congestion control algorithms. Motivated by the widespread deployment of highspeed links, many new aggressive congestion control protocols have been recently proposed [14, 13]. Such protocols are known to be very aggressive when competing with standard TCP, and in such environments, it is likely that some form of forced fairness is necessary to protect less aggressive flows. Thus, if we assume networks with heterogenous end-to-end congestion control strategies, the only way to control fairness in the network is to do it at routers using some form of queue management.

The design of router queue management algorithms has been a very active research area during last decade. Typically, a number of separate strategies can be discerned when reviewing this work; those strategies that use a single queue and do not require the router to process state information; those that require the router to process some state information and/or may utilize multiple router queues; and those that require changes to existing IP infrastructure. We use the term (AQM) to refer to the first of these, Fair Queueing to refer to the second of these, and New Architecture Proposals to refer to the last of these.

(1) Active Queue Management(AQM) : Traditional AQM's [2, 3, 4, 5] have been designed with the ultimate goal of providing high link utilization, low queueing delays, and small number of packet losses. A key issue in the design of AQM schemes is the complexity of their implementation and the requirement that they should be implementable on a wide range of Internet routers. As most AQM schemes do not differentiate packets from different flows, bandwidth allocation among users is mainly determined by end-to-end congestion control policies. For example, over routers with constant loss probability, the amount of bandwidth that standard TCP user obtain is proportional to inverse of its round-trip time[24, 23].

(1a) CHOKe : CHOKe is an example of an AQM that differentiates packets from different flows; flows with higher bandwidth are punished more than in RED or BLUE for example. This has the effect of improving fairness amongst responsive users, and the degree of improvement for some AQM schemes based on the basic Choke idea will be examined in Section 2

(2) Fair Queueing : Fair queueing mechanisms, like FQ, DRR or FRED [6, 7, 8], require the router to maintain some per-flow state information and to use this information to manage arriving packets. Although they achieve excellent fairness, they are not widely deployed in real networks, mainly because high computational costs associated with obtaining, processing and managing this state information.

(3) New Architecture Proposals : Several new proposals, including CSFQ[9], XCP[10] and MaxNet[11], require changes to the packet IP header. Although they perform quite well in terms of fairness and resource utilization, they assume cooperation of most (if not all) routers in the entire network which is clearly quite restrictive.

In this paper we will develop Active Queue Management(AQM) schemes that can help less aggressive users to achieve their fair[1] share of bandwidth without keeping explicit per-flow state information. To do so, one has to differentiate between packets from different flows. We shall see that probabilistic CHOKe-like techniques are extremely effective in doing just this and in developing AQM's that serve to protect less aggressive flows.

[1]Throughout this paper, we use term fair share to refer to well known max-min fair bandwidth allocation, see [21]

In Section 2 we extend the basic idea from CHOKe and propose a new AQM called Multi-level comparison (MLC). We shall see that MLC belongs to a class of AQM that are CHOKe-like. A given AQM is said to be CHOKe-like (CL) if a router drops a packet from a flow with current throughput U with probability $\rho_0 U^{l-1}$, where ρ_0 is variable controlled by router and l is a positive integer that is called the index of the CL scheme. We propose a model that captures fairness properties of network of TCP users and routers operating CL AQM's with an any given index l. In particular, we use this model to prove the following results.

CL-AQM: Single bottleneck case. For N TCP users with heterogenous RTT's traversing a single bottleneck router that employs a CHOKe-like AQM with index l, we show (Theorem 2.2) that the asymptotic throughput of the i'th network flow can be approximated by

$$\bar{U}_i \approx C_{CL} \frac{1}{RTT_i^{\frac{2}{l+1}}}, \qquad (1)$$

where $\bar{U}_i$ denotes the time-averaged throughput of the i'th flow and C_{CL} is a constant that is independent of i.

CL-AQM: General network case. For a given network of TCP users and routers operating CL-AQM's, each with index l, the share of bandwidth amongst the TCP users can be made arbitrary close to the max-min fair share by choosing l to be large enough (Theorem 2.3).

MLC appears to be difficult to implement for values of index greater than 2. In order to validate our mathematical results for general CL-AQMs with large values of l, we follow the approach of Core Stateless Fair Queueing (CSFQ) and describe another CHOKe-like scheme, that assumes information on throughput is explicitly written in packet header.

Both MLC and CC have a more theoretical than practical value: MLC because its high computational cost, and CC because it requires an additional field in the IP header. In Section 3 we develop another AQM called MAY (Markov Active Yield) that does not suffer from same complexity issues that are inherent in MLC nor requires changes in IP infrastructure. Briefly, MAY compares each packet that arrives at the queue with a number of random packets that are drawn from a historical record of previous packet drops. Based on this comparison(s), a decision is made and the arriving packet is either dropped or enqueued. To study fairness properties of TCP, and more eneral AIMD users, over a link that operates MAY AQM, we develop a stochastic model and under some assumptions we prove the following.

MAY, Single bottleneck case. For N-TCP users with heterogenous RTT's traversing a link with a MAY queue, the asymptotic throughput of the i'th network flow can be approximated by

$$\bar{U}_i \approx C_{MAY} \tag{2}$$

where $\bar{U}_i$ again denotes the time-averaged throughput of the i'th flow, and C_{MAY} is a constant that is independent of i. Furthermore, we also show that (2) holds in this case irrespective of the AIMD parameters of the individual flows. This property of MAY is probably the most surprising theoretical result of the present paper.

Although we are unable to develop a model that would capture fairness properties in general network topologies, and therefore are unable to prove formal results on bandwidth allocation for general networks with MAY queues, our experiments suggest that MAY enforces bandwidth allocation that is to max-min among AIMD users in general network topologies. In Section 4 we present packet level simulation results that validate our theoretical results using ns-2.

2 CHOKE-like AQM schemes

It has been noticed in many studies that both drop tail and RED routers have large bias against large-RTT flows. For example Lakshman&Madhow [16] have made the empirical observation that for a drop-tail router and two flows with round trip times RTT_1 and RTT_2, the ratio of asymptotic throughput of the first and the second flow is in the ratio $(RTT_2/RTT_1)^a$ for some $a \in (1, 2)$. Similarly, it has also been noticed in a number of studies, that *RED*-like AQM schemes (RED [2], BLUE [3]) which attempt to estimate the loss probability for a given traffic pattern and to drop packets according to this estimation, share bandwidth among competing users with round trip times RTT_1 and RTT_2 in the ratio RTT_2/RTT_1, [24, 23]. In this section we will investigate RTT unfairness characteristics for more general AQM schemes. In particular, we shall characterise the fairness properties of CHOKe-like AQM schemes.

Definition 2.1. *An AQM is CHOKe-like if it drops a packet from a flow with current throughput*[2] *U with probability $\rho_0 U^{l-1}$, where ρ_0 is variable controlled by router and l positive integer called index of given CHOKe-like AQM.*

Comment : With $l = 1$ this corresponds to a router which drops packets with loss probability ρ_0, exactly as BLUE in steady state. The case when $l = 2$ is similar to CHOKe in the limit when the average queue size does not go the below minimum threshold, and in addition, when there is neither a RED nor an overflow drop. Indeed, comparing a packet at the entrance of queue with a packet from the queue and making drop-decision based on this comparison is actually dropping a packet with probability which is proportional to current throughput of the flow.

In this section we will describe a class of CHOKe-like AQM's called Multi Level Comparison (MLC) AQM's. In particular, we will describe and analyse the fairness characteristics of this queueing discipline for TCP flows competing for bandwidth.

Definition 2.2. *Let $f : R^+ \to R^+$. We say that an AQM scheme is f-RTT-fair if two flows with round trip times RTT_1 and RTT_2 which have single bottleneck operating with given AQM, obtain bandwidth in ratio: $f(RTT_1)/f(RTT_2)$.*

We will see that the MLC scheme with index l achieves $1/RTT^{1/(l+1)}$-fairness under the assumption of low loss probability. More generally, Theorem 2.3 shows that increasing l leads to fairness among TCP users arbitrary close to max-min in general network topologies.

2.1 Description of MLC

The basic strategy in MLC is to develop the core idea from CHOKe of comparing of a packet arriving at the queue with packets which are already in queue; these stored packets are a measure of the proportion of bandwidth used by certain flow. In MLC, each packet is compared with $l - 1$ other randomly chosen packets from the queue; if all l packets are from same flow, the arriving packet is dropped, otherwise it is enqueued. MLC also maintains a variable h_M which is used to control the probability of dropping an arriving packet. In particular, if one trial represents a l-level comparison, h_M is the number of trials that are conducted for each arriving packet. Consequently, if the link is underutilized h_M should be decreased in order to decrease probability of dropping packets, on another hand if aggregate traffic on link is grater then link capacity, h_M should be increased.

At this point it is important to emphasize two crucial differences between MLC and CHOKe. First, note that CHOKe makes a comparison only when link is already congested (when average queue size becomes greater than min_{th}), and therefore its performance depends mainly on number of users. Clearly, a small number of users will affect synchronization of losses, while for large number of users, the number of CHOKe-drops will be much less then number of RED-drops and therefore the effect of CHOKe will be negligible. Second, in MLC, if a packet is not dropped at the entrance of the queue it will not be dropped latter while in the queue. On the other hand original CHOKe algorithm allows dropping both arriving packet and packet(s) from the queue.

Controlling the variable h_M : MLC uses a parameter $SampleTime$ to affect changes in the variable h_M (in our simulations this is set to $100ms$). h_M is adjusted once per $SampleTime$ using a MIMD (Multiplicative Increase - Multiplicative Decrease) scheme; see Figure 1 . Namely, if within the previous $SampleTtime$ the link was idle, h_M is set to γh_M for some $\gamma \in (0, 1)$, otherwise h_M is adjusted as $h_M := h_M \delta$ for some $\delta > 1$. The identification of idle periods is determined by weighted average queue size avg_q. Thus, if within one period $avg_q < min_{th}$ at all times, then

[2]Throughput is measured in packets per unit of time.

```
If (now - last_update)>SampleTime
    if (Link was idle)
        h_M := h_M*gamma
        last_update = now
    otherwise
        h_M := h_M*delta
        last_update = now
    end
end
```

Figure 1: Control of h_M

we assume that this period is idle, otherwise we assume that the aggregate traffic is greater than link capacity[3].

Comment : Our method of controlling h_M is very similar to way BLUE controls its drop-probability. For $l = 1$ MLC h_M actually represents a drop probability and in this case only differences between MLC and BLUE are the following: (1) MLC controls h_M with MIMD algorithm while BLUE uses AIAD scheme to control drop probability and (2) MLC uses queue-average while BLUE uses actual queue length as congestion indicator.

Our experiments indicate that the parameter $SampleTime$ should be in the range of round trip times from connections utilizing the link (in order to allow users to react to changes in h_M). The parameters γ and δ control the speed of adaptation to changes in network traffic and are set such that for a given $SampleTime$, h_M can be doubled/halved within a few seconds.

Sometimes, several users can stop using the link simultaneously. In order to ensure quick 'convergence' to the high utilization regime, MLC will not drop any packets if link utilization is less then *pct* percents. In our experiments, *pct* is set to 98%.

2.2 Model and analysis of CHOKe-like AQM

In this Section we present a model of CL-AQM's servicing multiple TCP users. We present several theorems that characterise this situation for both single bottleneck and general network topologies.

Single bottleneck case: We consider N TCP-flows with heterogenous round trip times RTT_i, $i = 1, \dots, N$, traversing a single bottleneck link that employs CL-AQM with an index l. If we assume that ρ_0 does not fluctuate much so that we can model it as constant and that the drop probability for a packet is small, then our analysis shows that the asymptotic rates achieved by TCP users are proportional to $\frac{1}{RTT_i^{2/(l+1)}}$. This is the main result of this section and is given in Theorem 2.2.

Model : At the flow level, let Δ to be length of sampling interval over which we evaluate changes in throughput. If a flow with round trip time RTT does not see a drop within interval of length Δ, then its throughput will be increased for Δ/RTT^2. If a flow registers a drop within this sampling interval then its throughput will be halved[4]. The probability that the first event will happen is equal to probability that each of ΔU packets from the flow are not dropped. This probability is given by:

$$\eta_1 = (1 - \rho_0 U^{l-1})^{\Delta U} \approx e^{-\Delta \rho_0 U^l}.$$

Clearly, the probability that a flow with current throughput U will see a drop within a sampling interval of length Δ is equal to

$$\eta_2 = 1 - (1 - \rho_0 U^{l-1})^{\Delta U} \approx 1 - e^{-\Delta \rho_0 U^l}.$$

The previous approximations are valid under the assumption of a small probability that a packet will be dropped : $\rho_0 U^{l-1} << 1$. This assumption seems reasonable: if this probability is not small, a flow would suffer too many losses an therefore would not get chance to grow.

Let $U_k^{(\rho)}$ be a stochastic process which describes evolution of throughput of a TCP flow with round-trip time RTT traversing over link with a CHOKe-like AQM scheme with index l. Here $\rho = \Delta\rho_0$. Since Δ is fixed we can assume that Δ to be equal to one unit of time.

We model $U_k^{(\rho)}$ as a Markov chain on $[0, \infty)$ defined by $U_0^{(\rho)} = 0$ and:

$$U_{k+1}^{(\rho)} = U_k^{(\rho)} + \frac{1}{RTT^2} \quad \text{with probability} \;\; e^{-\rho (U_k^{(\rho)})^l}$$

$$U_{k+1}^{(\rho)} = \frac{1}{2} U_k^{(\rho)} \quad \text{with probability} \;\; 1 - e^{-\rho (U_k^{(\rho)})^l}.$$

Let s_k be the time of k-th loss, ie: s_k is the k-th smallest m which satisfies $U_m^{(\rho)} = \frac{1}{2} U_{m-1}^{(\rho)}$. Define $Z_0^{(\rho)} = 0$ and $Z_k^{(\rho)} = U_{s_k}^{(\rho)}$. It follows that $Z_k^{(\rho)}$ is also a Markov chain and the following theorem provides insight into the behavior of $Z_k^{(\rho)}$ for small values of ρ.

Proposition 2.1. *There exist a Markov chain $\overline{V}_k$, such that for $\rho > 0$,*

$$Z_k^{(\rho)} = \frac{1}{RTT^{\frac{2}{l+1}} \rho^{\frac{1}{l+1}}} \overline{V}_k + \frac{1}{\rho^{\frac{1}{l+1}}} R_k^{(\rho)},$$

where the stochastic process $R_k^{(\rho)}$ converge to zero in distribution as $\rho \to 0$.

Preamble to Proof of Proposition 2.1

The Markov chain $U_k^{(\rho)}$ can be written as $U_k^{(\rho)} = RTT^2 W_k^{(\alpha)}$, where $\alpha = \rho / RTT^{2l}$ and

$$W_{k+1}^{(\alpha)} = W_k^{(\alpha)} + 1 \quad \text{with probability} \;\; e^{-\alpha (W_k^{(\alpha)})^l}$$

[3]As congestion indicator one can use also actual queue length or measured aggregate throughput, but we have chosen average queue in our simulations

[4]Throughout this paper, variations in round trip times are neglected.

$$W_{k+1}^{(\alpha)} = \frac{1}{2}W_k^{(\alpha)} \quad \text{with probability} \;\; 1 - e^{-\alpha(W_k^{(\alpha)})^l}.$$

Analogously, we can define another associated Markov chain with states of $W_k^{(\alpha)}$ just after congestion: $V_k^{(\alpha)} = Z_k^{(\rho)}/RTT^2$. Its transition is given by

$$V_{k+1}^{(\alpha)} = \frac{1}{2}(V_k^{(\alpha)} + G_{V_k^{(\alpha)}}^{(\alpha)}).$$

Here $G_n^{(\alpha)}$ is random variable determined by distribution:

$$P(G_x^{(\alpha)} \geq y) = \prod_{k=x}^{x+\lfloor y \rfloor} e^{-\alpha k^l}.$$

Lemma 2.1. *Let $x \geq 0$. Then as α goes to 0, the random variable $\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)}$ converges in distribution to a random variable $\overline{G}_x$ such that for $y \geq 0$*

$$P(\overline{G}_x \geq y) = e^{-\frac{1}{l+1}((x+y)^{l+1} - x^{l+1})}. \qquad (3)$$

Proof. Let $x, y \geq 0$ and $\alpha < 1$. Then

$$P(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y) = P(G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y/\alpha^{\frac{1}{l+1}})$$

$$= \prod_{k=x/\alpha^{\frac{1}{l+1}}}^{\lfloor (x+y)/\alpha^{\frac{1}{l+1}} \rfloor} e^{-\alpha k^l}.$$

Taking logarithms we conclude:

$$ln(P(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y)) = -\alpha \sum_{k=x/\alpha^{\frac{1}{l+1}}}^{\lfloor (x+y)/\alpha^{\frac{1}{l+1}} \rfloor} k^l \longrightarrow$$

$$\longrightarrow -\alpha \int_{x/\alpha^{\frac{1}{l+1}}}^{(x+y)/\alpha^{\frac{1}{l+1}}} t^l dt = \frac{1}{l+1}((x+y)^{l+1} - x^{l+1}).$$

□

Definition 2.3. *The sequence $\overline{V}_n$ denotes a Markov chain given by $\overline{V}_0 = 0$ and transitions:*

$$\overline{V}_{n+1} = \frac{1}{2}(\overline{V}_n + \overline{G}_{\overline{V}_n}).$$

Here $\{\overline{G}_x : \; x \geq 0\}$ is a family of random variables independent of $\overline{V}_n$ such that the distribution of $\overline{G}_x$ is given by (3).

Lemma 2.2. *The Markov chain $\alpha^{1/(l+1)}V_n^{(\alpha)}$ converges in distribution to the Markov chain $\overline{V}_n$.*

Proof. The proof of this fact follows same lines as proof of Corollary 1 in paper [17]. Namely, using uniform continuity of the mapping $t \rightarrow t^l$, and techniques from the proof of Proposition 1 from [17], it follows that for any $K > 1$

$$\lim_{\alpha \to 0} \sup_{\alpha^{1/(l+1)} < x, y < K} |P(\overline{G}_x \geq y) - P(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)} > y)| = 0.$$

Then, using the previously established uniform convergence, it follows that for any continuous function f on R^+ with compact support:

$$\lim_{\alpha \to 0} \sup_{x > \alpha^{1/(l+1)}} |E(f(\overline{G}_x)) - E(f(\alpha^{\frac{1}{l+1}} G_{x/\alpha^{\frac{1}{l+1}}}^{(\alpha)}))| = 0.$$

Finally, again following the same lines as in proof of Proposition 2 from [17] we can conclude desired convergence in distribution. □

The previous lemma allows us to approximate the Markov chain $V_k^{(\alpha)}$ with $\overline{V}_k$ with initial value $\overline{V}_0 = \alpha^{1/(l+1)}V_0^{(\alpha)} = 0$. We are ready to give:

Proof of Proposition 2.1. Recall that $Z_k^{(\rho)}$ represents throughput just after packet loss and that $\alpha = \rho/RTT^{2l}$. Then for small ρ we can write[5]

$$Z_k^{(\rho)} = \frac{1}{RTT^2}V_k^{(\alpha)} = \frac{1}{RTT^2}(\frac{1}{\alpha^{1/(l+1)}}\overline{V}_k + o(\frac{1}{\alpha^{\frac{1}{l+1}}})) =$$

$$= \frac{RTT^{2l/(l+1)}}{RTT^2}\frac{1}{\rho^{1/(l+1)}}\overline{V}_k + o(\frac{1}{\rho^{\frac{1}{l+1}}}) =$$

$$= \frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}}\overline{V}_k + o(\frac{1}{\rho^{\frac{1}{l+1}}})$$

□

Thus the Markov chain $Z_k^{(\rho)}$ of throughput after congestion events can be approximated with the Markov chain $\frac{1}{RTT^{\frac{2}{l+1}}\rho^{\frac{1}{l+1}}}\overline{V}_k$ for small values of ρ as was claimed.

At this point we are concentrating on $\overline{V}_k$ and following the basic ideas from [17]. We shall prove that $(\overline{V}_k)^{l+1}$ is autoregressive(AR) process with unique invariant distribution that can be explicitly written.

Theorem 2.1. *For the Markov chain $\overline{V}_k$, the process $\overline{V}_k^{l+1}$ is autoregressive with following representation:*

$$\overline{V}_{k+1}^{l+1} = \frac{1}{2^{l+1}}(\overline{V}_k^{l+1} - (l+1)E_n)$$

where E_n is an IID sequence of exponentially distributed random variables with parameter 1: $P(E_n \geq y) = e^{-y}$. Moreover the Markov chain $\overline{V}_k$ has unique invariant distribution represented by the random variable $\overline{V}_\infty$ which satisfy:

$$\overline{V}_\infty = \sqrt[l+1]{(l+1)\sum_{n=1}^{\infty}\frac{1}{2^{n(l+1)}}E_n}, \qquad (4)$$

and the process defined with this invariant distribution as initial distribution for $\overline{V}_0$ is ergodic.

[5]Here, $o(\frac{1}{\alpha^{\frac{1}{l+1}}})$ represent random variable $R(\alpha)$ such that $R(\alpha)/\alpha^{\frac{1}{l+1}} \to 0$ as $\alpha \to 0$

Proof. By definition we have

$$\overline{V}_{n+1} = \frac{1}{2}(\overline{V}_n + \overline{G}_{\overline{V}_n}).$$

Let $E_n = \frac{1}{l+1}(2^{l+1}\overline{V}_{n+1}^{l+1} - \overline{V}_n^{l+1})$. Then:

$$P(E_n \geq y) = P(\frac{1}{l+1}(2^{l+1}\overline{V}_{n+1}^{l+1} - \overline{V}_n^{l+1}) \geq y) =$$

$$= P((\overline{V}_n + \overline{G}_{\overline{V}_n})^{l+1} \geq (l+1)y + \overline{V}_n^{l+1})$$

$$= P(\overline{G}_{\overline{V}_n} \geq ((l+1)y + \overline{V}_n^{l+1})^{1/(l+1)} - \overline{V}_n) =$$

$$= e^{-\frac{1}{l+1}((l+1)y + \overline{V}_n^{l+1} - \overline{V}_n^{l+1})} = e^{-y}$$

Thus, the first part of the Theorem is proved. To prove the second part notice that random variable given by (4) represents an invariant probability. To prove its uniqueness we use Theorem 7.16 from [18]. It is enough to prove that there are no two disjoint sets $A_1, A_2 \subset R^+$ such that for $i = 1, 2$ and all $x \in A_i$, $P(\overline{V}_2 \in A_i | \overline{V}_1 = x) = 1$. Suppose that there exist two such sets A_1, A_2. Let $x_1 \in A_1$. Then since G_{x_1} has positive density we conclude that $[\frac{x_1}{2}, \infty) \setminus A_1$ has Lebesgue-measure 0. Similarly for $x_2 \in A_2$, $[\frac{x_2}{2}, \infty) \setminus A_2$ has Lebesgue-measure 0 which means that A_1 and A_2 cannot be disjoint. □

Now we are ready to prove that for small values of ρ_0, the steady state throughput can be approximated by $\frac{1}{RTT^{\frac{2}{l+1}} \rho_0^{\frac{1}{l+1}}} D_{CL}$, where D_{CL} does not depend on ρ_0 nor l.

Theorem 2.2. *The time averaged throughput of the i'th flow:* $\frac{1}{M}\sum_{i=1}^{M} U_i^{(\rho)}$ *converges almost surely to:*

$$\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M} U_i^{(\rho)} =: \overline{U}^{(\rho)} = \frac{1}{RTT^{\frac{2}{l+1}} \rho^{\frac{1}{l+1}}} D_{CL} + \frac{1}{\rho^{\frac{1}{l+1}}} S^{(\rho)}$$

where $D_{CL} = \frac{3}{4}\frac{E(\overline{V}_\infty^2)}{E(\overline{V}_\infty)}$ *and the constant* $S^{(\rho)}$ *converges to 0 as* ρ *goes to 0.*

Proof. If we write $U_i^{(\rho)} = RTT^2 W_k^{(\alpha)}$, where $\alpha = \rho/RTT^{2l}$, the Theorem is an immediate consequence of the Theorem 2.1 and Proposition 9 from [17] for a multiplicative decrease factor $\delta = \frac{1}{2}$.[6] □

To conclude this section we will prove that for a given network with routers employing a CL AQM with index l, and assuming that the steady state throughput is given by the previous theorem, we can find large enough l such that bandwidth allocation is arbitrary close to max-min fairness. The following characterization of max-min fairness can be found in [21].

Lemma 2.3. *A set of rates* x_r *is max-min fair if and only if for every flow r there exists a link on its path, such that the rates of all flows which traverse through that link are less or equal then* x_r.

[6]Here constant $\frac{3}{4} = \frac{1+\delta}{2\delta(1-\delta)}$, for $\delta = 0.5$.

With this characterization of max-min fair allocation in mind, we shall prove that increasing the index of the CL scheme will result in allocation of bandwidth in such fashion that each flow will have link on its path such that its asymptotic rate is 'almost' the largest among all flows using that link.

Theorem 2.3. *For any given network topology, and given* $\varepsilon > 0$, *there exist* l *such that if all queues employ CL AQM with index* l *and loss probabilities are small then for every flow r there exist a link on its path, such that the rates of all flows which traverse through that link are less then* $(1+\varepsilon)x_r$ *(here* x_r *is steady state rate of flow r).*

Proof. Let L be the number of links in the network and N the number of flows. We label flows by $i = 1, 2, \ldots, N$ and links by $s = 1, 2, \ldots, L$. By R we denote the routing matrix: $R_{is} = 1$ if flow i uses link s otherwise $R_{is} = 0$. On each link s, a router drops a packet from the flow with current throughput U with probability $\rho(s)U^{l-1}$. Let M be the length (in number of links) of the path of the flow with most links on its route and ν ratio of largest and smallest round trip time in the network. Choose l such that

$$\nu^{\frac{2}{l+1}} M^{\frac{1}{l+1}} < 1 + \varepsilon.$$

For each flow r, let $s_1^{(r)}, \ldots, s_w^{(r)}$ be links used by it and let $s_{max}^{(r)}$ the most congested link on its route in the following sense:

$$\rho(s_{max}^{(r)}) = \max\{\rho(s_j^{(r)}) |\ j = 1, \ldots, w\}. \quad (5)$$

If the current rate of flow r is U, a packet from that flow will be dropped with probability $\lambda_r U^{l-1}$, where $\lambda_r = \sum_{j=1}^{w} \rho(l_j^{(r)})$, and therefore the steady state throughput for flow r is given by

$$x_r = \frac{1}{RTT_r^{\frac{2}{l+1}} \lambda_r^{\frac{1}{l+1}}} C_0.$$

For any other flow t which uses the link $s_{max}^{(r)}$ with

$\lambda_t = \sum_{j:R_{jt}=1} \rho(l_j) \geq \rho(s_{max}^{(r)})$

the steady state throughput is given by:

$$x_t = \frac{1}{RTT_t^{\frac{2}{l+1}} \lambda_t^{\frac{1}{l+1}}} C_0.$$

Recall that we have defined the link $s_{max}^{(r)}$ as the most congested link on route of flow r in the sense of (5). This implies that $\lambda_r \leq M\rho(s_{max}^{(r)})$. Now

$$\frac{x_t}{x_r} = (\frac{RTT_r}{RTT_t})^{\frac{2}{l+1}} \frac{\lambda_r}{\lambda_t} \leq (\frac{RTT_r}{RTT_t})^{\frac{2}{l+1}} (\frac{M\rho(s_{max}^{(r)})}{\rho(s_{max}^{(r)})})^{\frac{1}{l+1}} \leq$$

$$\leq \nu^{\frac{2}{l+1}} M^{\frac{1}{l+1}} < 1 + \varepsilon$$

□

It is interesting to notice that the fairness of well known eXplicit Congestion Control - XCP, features a similar property. Namely, in the recent paper [15] the authors proved that "... *given any network topology, one can choose a shuffling parameter, γ, sufficiently small so that the resulting allocation is close to max-min fairness. For any fixed $\gamma > 0$, however, there are topologies in which some flow rates can be far away from their max-min allocations"*. In the case of CL schemes for given network topology we can chose sufficiently large index l so that resulting allocations are close to max-min, but for any fixed l there are topologies in which some flow rates can be far away from their max-min allocations.

2.3 CC-l, a version of CHOKe-like scheme with additional field in IP header

As we noticed earlier, MLC becomes extremely inefficient when its index becomes greater than 2. On the another hand if we use the basic idea from CSFQ [9] in which core routers have explicit information on current throughput for each flow, then the router can explicitly control ρ_0 and drop packets according to Definition 2.1. Recall that controlling h_M in MLC actually implicitly determines ρ_0 and essentially the effects of MLC and CC are same in terms of fairness.

3 Markov Active Yield (MAY) AQM's

In the previous section we have seen that the procedure of comparing an arriving packet with packets from the queue (or sampled list of recently seen packets), and making drop decision based on this comparison, can lead to fairness that is arbitrarily close to max-min fairness. A basic problem with this strategy is that it can be computationally very expensive to implement and it is therefore of interest to examine other methods of enforcing max-min fairness.

In order to enforce fairness among users with different levels of aggressiveness, and responsiveness, it is clear that more aggressive and less responsive users must be punished more than less aggressive and more responsive users. The question is how much and how should this be implemented in an efficient manner.

If an AQM does not differentiate among different flows, a packet drop will adversely affect much more responsive and less aggressive users. That is directly the source of unfairness among different flows. Our basic idea is to keep a list of recently dropped packets and to use the statistics from this list to regulate the dropping of packets.

Constructing the LIST : For each arriving packet we compare it with packet from a LIST of dropped packets of length L. If the LIST is not full, the arriving packet flow information[7] (SenderIP, sourceIP, protocolID) is added to it. The first $(1 - PctHistDrops)L$ elements of LIST are used to sample arriving traffic before packets are dropped or enqueued, and this information is written to a random position among the first $(1-PctHistDrops)L$ elements of the LIST. The last $PctHistDrops \cdot L$ elements of the LIST are reserved for the drop history. Thus, whenever a packet has to be dropped, its flow information is written to a random position among this $PctHistDrops \cdot L$ elements of the LIST while information written on that position is deleted. In our experiments, we use $PctHistDrops \in (0.9, 0.99)$.

[7] Any other definition for flow can be used.

Dropping packets. Each arriving packet is compared with h_{MAY} randomly chosen packets from the LIST. If one or more of these h_{MAY} packets belong to the same flow as the arriving packet, the arriving packet is then dropped (and therefore added to drop-history part of the LIST), otherwise it is enqueued. In either case, its flow information is written in the current-traffic part of the LIST. The variable h_{MAY} is then controlled by the router in order to achieve good utilization and keep the buffer from overflowing. In the event of a buffer overflow, dropped packets are added to the drop-list-history part of the LIST in sam way as they were dropped by MAY mechanism.

Control of h_{MAY}. The variable h_{MAY} determines how often the router should drop packets. The larger h_{MAY}, the more drops are experienced. This variable is adjusted once per $SamplingTime$. If within the last $SamplingTime$ interval the link was idle, then this event can be interpreted as too many packets have been dropped and h_{MAY} should be decreased. Otherwise, in the case of congestion, h_{MAY} should be increased in order to increase the probability of drops. In this implementation we are using MIMD (Multiplicative Increase - Multiplicative Decrease) algorithm for controlling h_{MAY}. Thus $h_{MAY} := h_{MAY}/t_2$ if the link is not congested at all during the previous $SamplingTime$ interval and $h_{MAY} := h_{MAY} \cdot t_1$ otherwise. Here $t_1 > 1$ and $t_2 > 1$ are the MIMD parameters. The identification of idle intervals can be done in several different ways (e.g. measuring the actual arrival rate, measuring instantaneous or average queue length). In the current implementation we use weighted average queue length, with weighted average constant q_w, as the congestion indicator. As soon as average queue goes above the $MinimumThreshold$ we identify the sampling interval as non-idle, otherwise, if the average queue length is smaller then $MinimumThreshold$ at all times within the sampling interval, we identify interval as an idle one.

Technical details. Sometimes several users can leave the network and the aggregate arrival rate may become significantly smaller than the link service rate. In order not to further underutilize the link we do not drop any packets if current arrival rate is less then $Pct\cdot$ (service rate of the link). In all our experiments Pct is set to 0.98. In parallel to this, whenever the arrival rate within last $SamplingTime$ interval is less then $Pct\cdot$(service rate of the link) we divide h_{MAY} not by t_2, but by $1+5(t_2-1)$ in order to improve 'convergence' to desirable values of h_{MAY}. On the other hand, if for the same reason, the buffer overflows, then we consider this as an indication of very high congestion and at the end of the sampling interval we multiply h_{MAY} not with t_1 but with $1 + 5(t_1 - 1)$.

Discussion of parameters. Essentially there are 5 parameters in MAY: t_1, t_2, $SamplingTime$, size of the LIST L, and $PctHistDrops$ and all of them determine speed with which adapting the AQM parameters to variable traffic conditions takes place. The triple $(SamplingTime, t_1, t_2)$ are not independent and should be chosen in such a fashion that h_{MAY} can be halved/doubled within few seconds. In our simulations we use $SamplingTime = 0.1sec, t_1 = 1.01, t_2 = 1.01$. The parameters $PctHistDrops$ and L determines how quickly a new flow will be punished as much as old flow with same characteristics. Different degrees of punishment for old and new flows might be not entirely undesirable as most of the new flows may be short Web-traffic flows which should be protected more than long-lived flows. There is naturally a tradeoff between network fairness and the time needed for a new flow to become statistically significant in the LIST: the larger $PctHistDrops$ is, the closer the fairness will be to max-min fairness but the 'convergence' time will be longer.

3.1 Mathematics

Having described the main algorithm, we will now prove that under following assumptions, arbitrary AIMD flows (flows with arbitrary linear increase parameter and arbitrary multiplicative decrease parameter), sharing a single link asymptotically obtain equal amount of available throughput. Throughout this section we make the following assumptions.

Assumption 3.1. *There are N long-lived flows that use a congested link and all of them employ AIMD congestion control algorithms with an additive increase parameter $\gamma_i > 0$ packets per unit of time, and a multiplicative decrease $\delta_i \in (0,1)$, $i = 1, 2, \ldots, N$.*

Assumption 3.2. *The variable h_{MAY} is constant.*

Assumption 3.3. *The statistics of the flows in the LIST are built in such fashion that the probability that a randomly chosen packet from the LIST belongs to certain flow i is equal to constant λ_i.*

Assumption 3.4. *The queue does not overflow and all MAY drops are enforced by drop-history part of the LIST.*

The first assumption defines the type of congestion control algorithms employed by users on the link.

Recall that h_{MAY} determines the drop probability. So the second assumption is actually saying that in the interval of interest this drop probability do not change.

The third and fourth assumption are made in order obtain a tractable model for analysis. Note that under static network conditions, as result of statistical multiplexing, the proportion of drops from certain in the drop history part of the LIST is asymptotically constant. However, in our model we assume that it is constant at each instant of time. Finally, the assumption that all drops are enforced by the drop history part of the LIST is an approximation to the fact that the percentage of drop history part of the LIST, $PctHistDrops$, is close to 1.

Preamble to main result :

In order to precisely formulate the main result, we present a model of the evolution of throughput for the i'th flow traversing the link.

Let $U_k^{(i)}$ be the throughput of flow (i) measured after k units of time. Assuming (3.4), the probability that among randomly chosen h_{MAY} packets from the LIST there is a packet from flow (i) that is equal to $\lambda_i \eta$, where η is function of h_{MAY}. Since we have assumed that h_{MAY} is constant, we then have that η is constant too. Thus, the loss probability for a packet from flow (i) is constant and is equal to $\mu_i = \lambda_i \eta$. Having this, we can consider $U_k^{(i)}$ as Markov chain on R^+ given by the transition:

$$U_{k+1}^{(i)} = U_k^{(i)} + \gamma_i \quad \text{with probability} \;\; e^{-\mu_i U_k^{(i)}}$$

$$U_{k+1}^{(i)} = \delta_i U_k^{(i)} \quad \text{with probability} \;\; 1 - e^{-\mu_i U_k^{(i)}}.$$

We can scale the Markov chain $U_k^{(i)}$, such that additive increase is equal to 1: $W_k^{(i)} = (1/\gamma_i) U_k^{(i)}$.

$$W_{k+1}^{(i)} = W_k^{(i)} + 1 \quad \text{with probability} \;\; e^{-\theta_i W_k^{(i)}}$$

$$W_{k+1}^{(i)} = \delta_i W_k^{(i)} \quad \text{with probability} \;\; 1 - e^{-\theta_i W_k^{(i)}}.$$

Here $\theta_i = \gamma_i \mu_i = \gamma_i \lambda_i \eta$. In [17], techniques for the analysis of Markov the chain $W_k^{(i)}$ were developed. Employing these tools we will prove the main result of our paper:

Theorem 3.1. *Under assumptions (1-4), the asymptotic throughput of flow (i)*

$$T^{(i)}(\delta_i) = \lim_{m \to \infty} \frac{1}{m} \sum_{k=1}^{m} U_k^{(i)}$$

converges almost surely, and for small η, $T^{(i)}(\delta_i)$ do not depend on additive increase parameter γ_i or the multiplicative decrease parameter δ_i. Formally, there exist a constant $c > 0$ such that:

$$T^{(i)}(\delta_i) = \frac{1}{c\eta + o(\eta)} + o(\frac{1}{\sqrt{\eta}}) \tag{6}$$

Proof. Before we begin the proof, we introduce some notation from [17]. By $V_m^{(i)}$ we denote the Markov process with states of $W_k^{(i)}$ just after packet losses, i.e.: $V_m^{(i)} = W_{s_m}^{(i)}$, where s_m is the m-th smallest k such that $W_k^{(i)} = \delta_i W_{k-1}^{(i)}$.

It is clear that $V_k^{(i)}$ is Markov chain with transition given by:

$$V_{k+1}^{(i)} = \delta_i (V_k^{(i)} + G_{V_k^{(i)}}^{(i)})$$

where $G_x^{(i)}$ is random variable on positive integers defined by

$$P(G_x^{(i)} \geq m) = \prod_{k=x}^{x+m-1} e^{-\theta_i k}.$$

The following proposition is Corollary 1 from [17].

Lemma 3.1. *If the initial state of the Markov chain $V_k^{(i)}$ is 0, then as η goes to 0, the Markov chain $\sqrt{\gamma_i \lambda_i \eta} V_k^{(i)}$ converges in distribution to the Markov chain $\overline{V}_k$ defined by $\overline{V}_0 = 0$ with transition:*

$$\overline{V}_{k+1} = \delta_i (\overline{V}_k + \overline{G}_{\overline{V}_k}) \tag{7}$$

where $\overline{G}_x$ is a random variable given by the distribution $P(G_x \geq y) = e^{-\frac{1}{2}y^2 - xy}$.

From the Proposition 7 [17] we conclude that the Markov chain with transition given by (7), has a unique stationary distribution $\overline{V}_\infty$. From Proposition 5[17] we get that

$$E(\overline{V}_\infty) = \frac{\delta_i}{1-\delta_i} E(G_{\overline{V}_\infty}). \tag{8}$$

Further, from Theorem 1[17] and Lemma (3.1), we conclude that as η goes to 0, the invariant distribution of $\sqrt{\gamma_i \lambda_i \eta} V_m^{(i)}$ converges to $\overline{V}_\infty$ and therefore

$$E(G_{V_\infty^{(i)}}) = \frac{1}{\sqrt{\gamma_i \lambda_i \eta}} E(G_{\overline{V}_\infty}) + o(\frac{1}{\sqrt{\eta}}) \tag{9}$$

On the other hand $G_{V_\infty^{(i)}}$ determines the times between drops from the flow (i). Thus the asymptotic proportion of drops from the flow (i) in the LIST is determined by frequency of drops and therefore equal to

$$\lambda_i = \frac{c}{E(G_{V_\infty^{(i)}})} \tag{10}$$

for a constant c that is equal to average time between two consecutive drops at MAY link.

Now we are ready to conclude the proof of the Theorem. From the Proposition 9 from [17] we have almost sure convergence of asymptotic throughput to $T^{(i)}(\delta_i)$ and:

$$T^{(i)}(\delta_i) = \gamma_i \lim_{m\to\infty} \frac{1}{m} \sum_{k=1}^{m} U_k^{(i)}$$

$$= \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \eta}} \frac{\delta_i}{(1-\delta_i) E(\overline{V}_\infty)} + o(\frac{1}{\sqrt{\eta}}). \tag{11}$$

Combining (11) with (8),(9) and (10) we obtain:

$$T^{(i)}(\delta_i) = \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \eta}} \frac{\delta_i}{(1-\delta_i)\frac{\delta_i}{1-\delta_i} E(G_{\overline{V}_\infty})} + o(\frac{1}{\sqrt{\eta}})$$

$$= \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \eta}} \frac{1}{E(G_{\overline{V}_\infty})} + o(\frac{1}{\sqrt{\eta}})$$

$$\frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \eta}} \frac{1}{\sqrt{\gamma_i \lambda_i \eta} E(G_{V_\infty^{(i)}}) + o(\sqrt{\eta})} + o(\frac{1}{\sqrt{\eta}})$$

$$= \frac{\gamma_i}{\sqrt{\gamma_i \lambda_i \eta}} \frac{1}{\sqrt{\gamma_i \lambda_i \eta}\frac{c}{\lambda_i} + o(\sqrt{\eta})} + o(\frac{1}{\sqrt{\eta}}).$$

Thus we have

$$T^{(i)}(\delta_i) = \frac{1}{c\eta + o(\eta)} + o(\frac{1}{\sqrt{\eta}})$$

and the proof is finished. □

3.2 Discussion of MAY and its limitations

Fairness in general network topologies. We have observed empirically that the bandwidth allocation of AIMD users over topologies with multiple congested links operating MAY schemes is very close to being max-min fair. To compare how far bandwidth allocations $U = (U_1, \ldots, U_N)$ deviate from max-min fair bandwidth allocations, $U_{mm} = (U_{1,mm}, \ldots, U_{N,mm})$, one may use Jain's index the given by:

$$j(U) = \frac{\left(\sum_{i=1}^{N} \frac{U_i}{U_{i,mm}}\right)^2}{N \sum_{i=1}^{N} \left(\frac{U_i}{U_{i,mm}}\right)^2}. \tag{12}$$

Clearly, $j(U)$ has a global maximum 1 attained at $U = U_{mm}$ and since it is continuous, and by measuring how far the index os from 1, one can give us some intuition on how far is vector U from U_{mm}.

A number of simulations are given in the next section. For the second of these in Section 4, the following indices for DropTail, RED and MAY were obtained :

$$j(U_{DropTail}) = 0.35, \; j(U_{RED}) = 0.73, \; j(U_{MAY}) = 0.985.$$

We believe that such good fairness properties of MAY follows from the fact that the drop history at a congested link is made up mainly from packets from flows which get most bandwidth and therefore flows which are not bottlenecked at the link (with average rate less then maximal average rate at the link) experience very few drops at that link. This means that a large majority of packets dropped at the link are from the flows bottlenecked at that link. If we ignore all other flows in network, we are in a situation where all bottlenecked flows represent the single bottleneck case, and therefore results from the previous paragraph apply.

To illustrate the fact that flows with average rate less than fair share are afforded extra protection at a link employing MAY we performed the following experiment using the network simulator $ns-2$. We ran a simulation with 5 TCP flows: $f1, f2, f3, f4$ and $f5$ over a single congested link employing MAY. Flows $f1, f2, f3$ and $f4$ have an unlimited maximal window size ($maxcwnd$), while $maxcwnd$

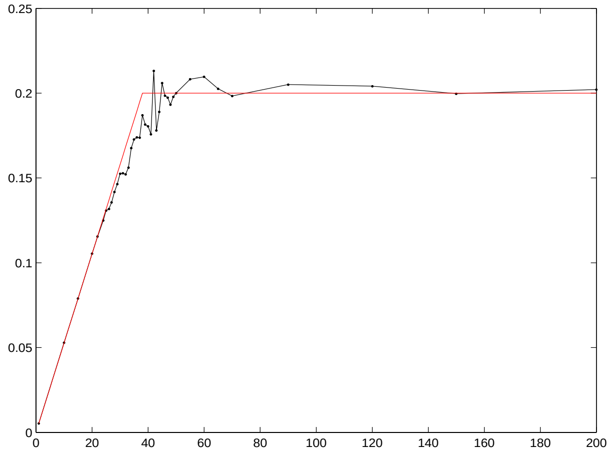

Figure 2: Maximal window size vs share of throughput

for flow $f5$ is varied from 1 to 200 packets. For a maximal window size 38, flow $f5$ would get approximately its max-min fair share of available bandwidth. As we can see from Figure 2, the throughput obtained by flow $f5$ in the presence of MAY drops for $maxcwnd$ from [1,38] is almost equal to throughput which would be achieved without any drops. For $maxcwnd \in [38, 200]$, MAY keeps the bandwidth allocated to flow $f5$ close its fair share of 20%.

Convergence. Building the LIST is the central task for ensuring farness using MAY. Building good statistics of the LIST is a consequence of the statistical multiplexing for long-lived flows. The following MAY parameters mainly determine the speed at which these statistics accumulate: $PctHistDrops$ and L-LIST size.

For $PctHistDrops = 0$, MAY behaves exactly same as MLC with index 2. For $PctHistDrops = 1$, there is no chance for new flow to join the LIST except at a buffer overflow. Buffer overflow is clearly not desirable since it would destroy the building of the list as a continuous process and it would punish flows without any control. Thus, we have to set $PctHistDrops$ to number strictly less then 1 in order to ensure building of the LIST in real time, but not much smaller than 1 so that the bandwidth allocation stays close to max-min fairness. This tradeoff: between convergence time and fairness, certainly depends on the nature of the traffic on the link. In situations where most of traffic is made up of short-lived flows, it is not possible to build a LIST fast enough to punish these in a desirable manner. However, in that case it is unclear what would be proper definition of fairness among short flows. On the other hand, in the environments where most of traffic is made up of long-lived flows, MAY will be able to control fairness of these long flows, and traffic represented by short flows will be protected from MAY. The size of the LIST, L, is another MAY parameter that affects the convergence time for building the LIST statistics. It should be set in range of $O(N)$ where N is number of *long flows* at link. This is related to Theorem 1 from [25] which claims that any algorithm that imposes max-min fairness with given relative error ε among N flows by converging to constant drop probabilities for each flow requires $O(N)$ space. However, greater values of L would give more precise statistics, but also result in longer convergence times. A size of L in range of $5N - 50N$ worked fine in our experiments.

RED parameters : q_{weight} and $MinimumTreshhold$, are used for identifying idle periods and should be chosen in such manner that MAY reacts quickly to a change in queue size, but also allows transient burst at the queue to be accommodated without initiating a harsh reaction from the AQM. In our experiments we use $q_{weight} = 1/($number of packets arrived in queue within one $SampleTime)$ and $MinimumTreshhold = 5 - 10\%$ of queue size. Of course, adaptive tuning of RED parameters may be applied as well as another methods for identifying idle periods.

Variable packet size : The problem of variable packet size can be easily solved by employing byte-wise comparisons instead packet-wise comparisons. This means that for each arrived packet, a variable that controls drop probability h_{MAY} should be multiplied by packet size.

Limitations of MAY : We have already noted certain problems related to effect of MAY in given traffic environment especially problems related to choosing of parameters. We believe that most of these problems can be resolved by some sort of continuous adapting of parameters and self tuning in order achieve certain objectives.

However there is the problem of number of comparisons that can be effectively done by processor at router. Namely, the number of comparisons needed by MAY might be larger then number of comparisons that can be done by processor at very high speed back bone routers (with speed greater than 1Gbps) processing a large number of long-lived flows. For example, consider a 10Gbps link used by 10 000 long-lived TCP flows, each with a packet size of 1000 bits and and RTT of 100ms. Each of them would get approximately $1.25{\cdot}10^5 B/sec = 125 pkt/sec = 12.5 pkt/RTT$. For an average rate of $12.5 pkt/sec$, approximately one out 100 packets should be dropped. Assuming that all flows are symmetric, the LIST will contain roughly same number of flow identifiers from each of these 10000 long flows. Thus in order to have loss rate of one out of hundred packets we have to compare the flow identifier of each arriving packet with approximately $10000{\cdot}1/100 = 100$ identifiers from the LIST. Since there are around $1.25 \cdot 10^6$ packet arrivals per second, the total number of MAY comparisons that should be done per second in this example is around $100 \cdot 1.25 \cdot 10^6 = 125 \cdot 10^6$. With more flows at the link and larger loss rate this number can be even larger. It might be argued that it is cheaper to keep per-flow state than doing so many comparisons. This argument depends mainly on flow size distribution and research in this direction remain to be done. However, we believe that heavy tail nature of the distribution of flow sizes [19, 20], where small number of flows account for large amount of traffic would be a key point in the ability of MAY to operates well in real networks even on high speeds.

Finally, the proposed design of MAY is constructed to control the fairness properties of responsive flows. In the presence of a non responsive UDP flow which sends more than its fair share, the drop history part of the list will eventually contain only identifiers of this UDP flow. Therefore, although the rate of this flow will be controlled by MAY, the LIST will not accurately reflect the statistics of the other flows in order to enforce fairness among other responsive TCP flows. One approach to this problem is to use MAY's LIST to identify nonresponsive users with rate greater than their fair share, then to employ a penalty box approach [22] and allow pollution of the LIST with identifiers from nonresponsive users. Note that the proportion of flows that are sending less than their fair share in the drop history of MAY is small, much less than in drop history of RED. This means that identifying flows with high rates from the drop history of MAY is easier that same from the drop history of RED.

4 Experimental results

In this section we briefly describe some experiments that demonstrate the efficacy of our proposed AQM schemes.

Single bottleneck topologies

The first set of experiments are designed to demonstrate the fairness properties of the proposed AQM schemes over single link. Specifically, we present results for a single link with service rate of $10MBps$ that services 100 long-lived TCP users with round trip times uniformly distributed in range $40 - 440ms$ and approximately $10ms$ of queueing delay. To provide baseline results, we include the performance of RED for the same scenario.

It can be clearly seen from our results that that the fairness of RED is approximately proportional to the inverse of RTT. This is in accordance with our theoretical results under assumption that drop probability is constant and is also consistent with results and observations made by other authors [23, 24]. It can also be observed that the fairness of MLC with index 2 is proportional to $1/RTT^{2/3}$ as predicted by Theorem 2.2. In our figures, * denotes the scaled model predictions, '+' denotes measured values of throughput, and the dotted line is estimate of the fair share. The MLC parameters used are in the experiment are: $index = 2$, $SamplingTime = 100ms$, $\gamma = 0.99$, $\delta = 1.01$, $Pct = 0.98$. The MAY parameters used in the experiment are: $SamplingTime = 100ms$, $t_1 = 1.01$, $t_2 = 1.01$, $Pct = 0.98$, $PctHistDrops = 0.98$, $L = 1500$.

In Figures 6 - 8 the link utilization of the link is evaluated for RED, MLC-1 and MAY. Our throughput measurements are averaged over $5sec$ intervals. For this experiment, with 100 flows both h_M and h_{MAY} are about 1. Which gives computational cost of MLC-1 and MAY of approximately 1 comparison per arrival.

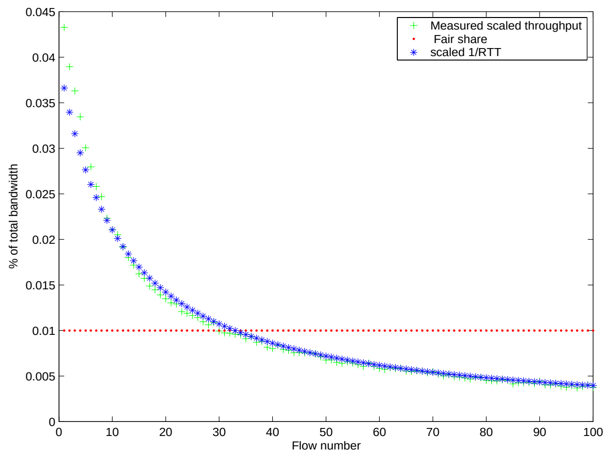

Figure 3: Scaled throughput for 100 flows over congested link employing RED

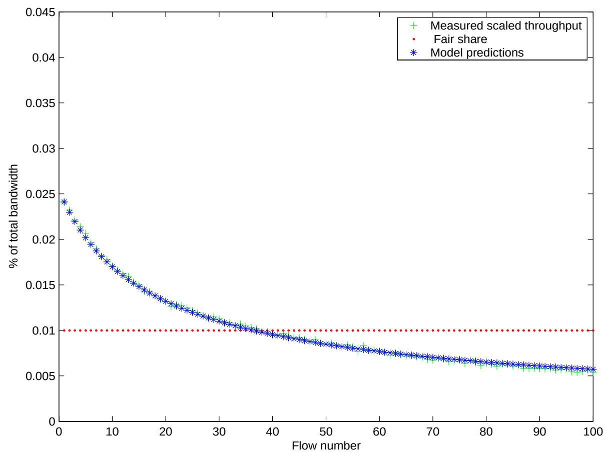

Figure 4: Scaled throughput for 100 flows over congested link employing CHOKe-like, MLC with index 2

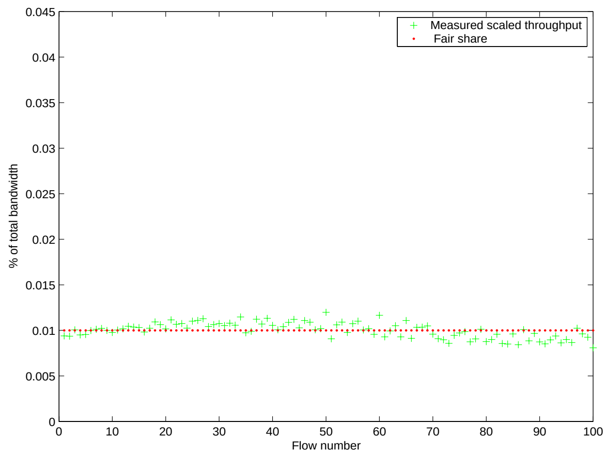

Figure 5: Scaled throughput for 100 flows over congested link employing MAY

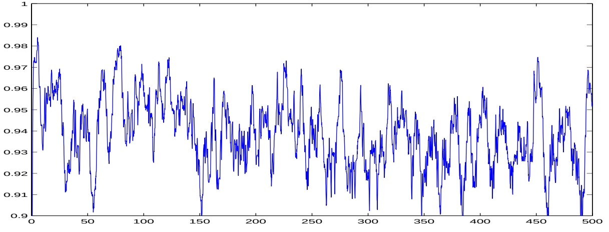
Figure 6: Link utilization with RED

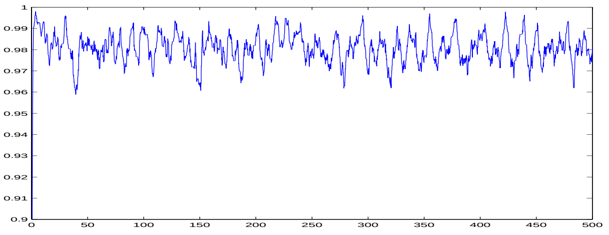
Figure 7: Link utilization with MLC-1

Multiple bottleneck topologies

Our second set of results demonstrate Theorem 2.3. The network topology that we considered is given in Figure 9. Here, we consider a network of 24 nodes: $n1-n5, m1-m5, p1-p5, q1-q5$, and $c1$, $c2$, $c3$, $c4$ and 30 flows traversing the network as follows: $n(i) \to p(i); n(i) \to q(i), m(i) \to p(i); m(i) \to q(i); n(i) \to m(i); p(i) \to q(i)$ where $i = 1, 2, 3, 4, 5$.

The delays on each of the links in ms are defined as follows:

$$\begin{aligned} ni \to c1 &: 40 * i + 1 \\ pi \to c3 &: 40 * i + 1 \\ mi \to c2 &: 40 * i + 1 \\ qi \to c4 &: 40 * i + 1 \end{aligned}$$

and the delays $c1-c2$, $c2-c3$, $c3-c4$ are $10ms$.

Each flow uses the standard TCP-SACK algorithm with delayed acking switched off, with a packet size 1000B. The aggressiveness of each flow is therefore mainly determined by its RTT. The behavior of the network is evaluated with each link $c1-c2$, $c2-c3$ and $c3-c4$ using: DropTail, RED, CC-1, CC-2, CC-4, CC-5, CC-9, CC-17 and MAY with a queue size 100 packets. CC-l parameters are: $index = l$, $SamplingTime = 100ms$, $\gamma = 0.99$, $\delta = 1.01$, $Pct = 0.98$. The MAY parameters used in the experiment are: $SamplingTime = 100ms$, $t_1 = 1.01$, $t_2 = 1.01$, $Pct = 0.98$, $PctHistDrops = 0.98$, $L = 300$.

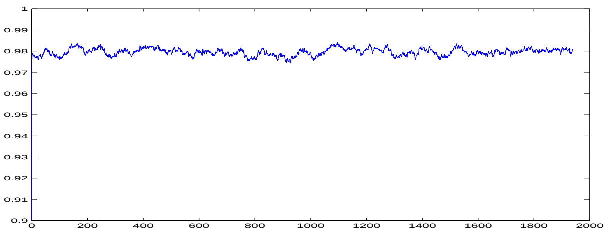
Figure 8: Link utilization with MAY

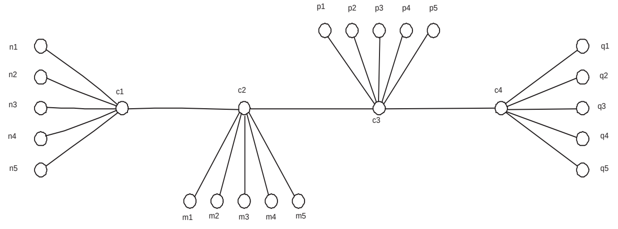

Figure 9: Network topology

Figure 10 depicts 9 scenarios, one for each dropping scheme. Each plot depicts the amount of throughput taken by each of 30 flows. The doted line represents the max-min fair share of bandwidth while points represent measured share of bandwidth for each of 30 flows in the network. We can see significant unfairness in DropTail, RED and CC-1 cases. As we increase index of CL scheme, we obtain share of bandwidth very close to max-min share as predicted by Theorem 2.3. Jain's indices, defined by (12) for each of these AQM's are:

$$j(U_{DrTail}) = 0.3498,\ j(U_{RED}) = 0.7310,\ j(U_{CC-1}) = 0.6612$$

$$j(U_{CC-2}) = 0.8462,\ j(U_{CC-3}) = 0.8848,\ j(U_{CC-5}) = 0.9597$$

$$j(U_{CC-9}) = 0.9892,\ j(U_{CC-17}) = 0.9970,\ j(U_{MAY}) = 0.9851$$

5 Conclusions and future work

In this paper we studied how fairness among TCP users is governed by choosing different AQM strategies at congested links. For that purpose we presented stochastic models that describe each of these AQM strategies. In particular, our principal concerns were with CHOKe-like AQM schemes that were defined in Section 2 and with the MAY AQM scheme defined in Section 3.

For CL schemes we showed that by increasing the CL-index, bandwidth allocations among TCP can be made arbitrarily close to the max-min fair allocation. Two approaches for design of CHOKe-like schemes were proposed: MLC and CC. While MLC requires very low memory space (of order O(1)) in order to accurately enforce max-min fairness it requires very large computational complexity. This tradeoff is a topic for further research. The CC AQM avoids such complexity issues but requires additional field in the packet header.

MAY is a simple AQM scheme with reasonable complexity requirements that enforces max-min fairness among responsive users without keeping per-flow states. For the MAY scheme we show that for the single bottleneck case, bandwidth allocation among arbitrary AIMD users is fair. Although we observed that bandwidth allocation are close to max-min also in general network topologies we were not able to obtain mathematical results in support of this observation. In Section 3.2 we suggested an empirical argument to explain this observations. Extending our model to this general network topologies would be important for fully understanding of MAY.

The problems discussed in Section 3.2 suggest that the choice of MAY parameters should be made carefully. Moreover, for certain links with a highly dynamic traffic distribution, it might be necessary to allow self tuning of parameters in order to achieve high utilization as well as good fairness among flows.

Finally, the inability of MAY to fully control nonresponsive flows is certainly very important issue that should be addressed in future work. This is general problem of most low-state AQM's that tries to enforce fairness, such as Stochastic Fair BLUE[3] or Stabilized RED [26]. In the Section 3.2 this problem is discussed and one possible approach is outlined.

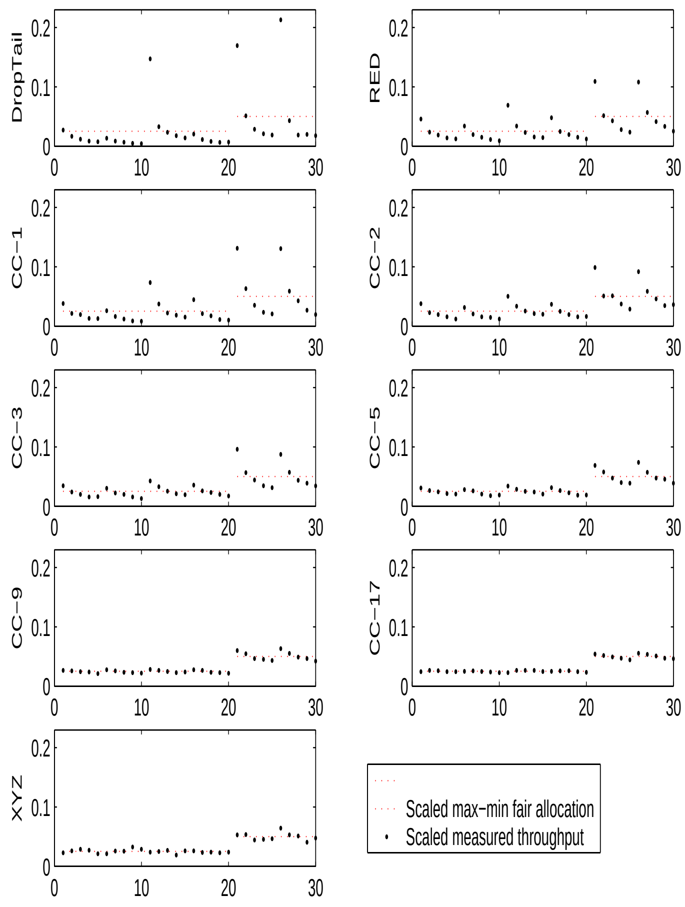

Figure 10: Amount of bandwidth taken by each of 30 flows. x-axis is flowID, y-axis is share of throughput. Droptail, RED, CL-1, CL-2, CL-3, CL-5, CL-9, CL-17, MAY

References

[1] V. Jacobson, Congestion avoidance and control, Proceedings of SIGCOMM, 1988.

[2] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking, vol. 1, pp. 397 -413, Aug. 1993.

[3] W. Feng, K.G. Shin, D.D. Kandlur, D. Saha. The BLUE active queue management algorithms. *IEEE/ ACM Transactions on Networking*, vol. 10, no. 4, 513-528, August 2002.

[4] C. Hollot, V. Misra, D. Towsley, W.B. Gong. Analysis and design of controllers for AQM routers supporting TCP flows *IEEE Transactions on Automatic Control*, pp. 945-959 June, 2002.

[5] S. Kunniyur, R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, 286-299, April 2004.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In Proceedings of the ACM SIGCOMM, Austin, TX, September 1989.

[7] M. Shreedhar, G. Varghese. Efficient fair queueing using deficit round-robin. *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, June 1996.

[8] D. Lin, R. Morris. Dynamics of random early detection. Proceedings of the ACM SIGCOMM '97, pp. 127 - 137.

[9] Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in

High Speed Networks. *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, 33-46, February 2003.

[10] D. Katabi, M. Handley, C. Rohr, Internet Congestion Control for Future High Bandwidth-Delay Product Environments, ACM Computer Communication Review Proceedings of the Sigcomm '02 Symposium, August 2002.

[11] B. Wydrowski, M. Zukerman, MaxNet: A congestion control architecture for maxmin fairness, IEEE Communications Letters, vol. 6, no. 11 , Nov. 2002, pp.512-514.

[12] Y. Yang, S. Lam. General AIMD Congestion Control, Proceedings ICNP 2000, Osaka, Japan, November 2000.

[13] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649, Experimental, December 2003.

[14] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks, Submitted for publication, December 2002

[15] S. Low, L. Andrew, B. Wydrowski. Understanding XCP: Equilibrium and Fairness. Proceedings of IEEE Infocom, Miami, FL, March 2005.

[16] T. V. Lakshman, U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, 336-350, June 1997.

[17] V. Dumas, F. Guillemin, P. Robert. A Markovian analysis of additive-increase multiplicative-decrease algorithms. Adv. in Appl. Probab. 34 (2002), no. 1, 85111.

[18] Leo Breiman, Probability, SIAM classics in Applied Math. 1992.

[19] M. Kodialam, T. V. Lakshman, Shantidev Mohanty. Run based Traffic Estimator (RATE): A Simple, Memory Efficient Scheme for Per-Flow Rate Estimation. Proceedings of IEEE INFOCOM, Hong Kong, March 2004.

[20] C. Estan, G. Varghese. New Directions in Traffic Measurement and Accounting. SIGCOMM, August 2002

[21] R. Srikant. The Mathematics of Internet Congestion Control. Birkhauser, 2004.

[22] K. Fall, S. Floyd. Router mechanisms to support end-to-end congestion control. [online] ftp://ftp.ee.lbl.gov/papers/collapse.ps.

[23] A. Abouzeid, S. Roy. Analytic understanding of RED gateways with. multiple competing TCP flows, in Proceedings of IEEE GLOBECOM, 2000.

[24] S. Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: Oneway Traffic, ACM Computer Communication Review, 30 - 47, Vol. 21 , Issue 5, October 1991.

[25] A. Das, D. Dutta, A. Goel, A. Helmy, J. Heidemann. Low State Fairness: Lower Bounds and Practical Enforcement. Proceedings of the IEEE INFOCOM, Miami, FL, USA, March 2005.

[26] T. J. Ott, T. V. Lakshman, L. H. Wong. SRED: Stabilized RED. Proceedings IEEE INFOCOM, New York, March 1999.

[27] B. Suter, T.V. Lakshman, D. Stiliadis, A.K. Choudhury. Buffer management schemes for supporting TCP in gigabit routers with per-flow queueing. IEEE Journal on Selected Areas of Communications 17 (6) (1999) 1159-1170.