Drop counters are enough

Rade Stanojević and Robert Shorten
Hamilton Institute, NUIM, Ireland

***Abstract*— Small Flow Completion Time (FCT) of short-lived flows, and fair bandwidth allocation of long-lived flows have been two major, usually concurrent, goals in the design of resource allocation algorithms. In this paper we present a framework that naturally unifies these two objectives under a single umbrella; namely by proposing resource allocation algorithm Markov Active Yield (MAY). Based on a probabilistic strategy: *"drop proportional to the amount of past drops"*, MAY achieves very small FCT among short-lived flows as well as max-min fair bandwidth allocation among long-lived flows, using only the information of short history of already dropped packets. It turns out that extremely small amount of on-chip SRAM (roughly 1 bit per flow in Pareto-like flow size distributions) is enough for storing this drop history. Analytical models are presented and analyzed and accuracy of results is verified experimentally using packet level $ns2$ simulations.**

I. Introduction

Resource allocation in communication networks has been a major topic of interest for some time. Many current proposals have as their performance objective a bandwidth allocation that is max-min fair [25], [5], [20], [11], [2], [26]. However, from the user point of view, a major (arguably the most important) performance metric is Flow Completion Time (FCT) [6]. A number of schemes that minimizes FCT exist in the literature [19], [3], [6], [10]. These require either state for every arriving flow, or cooperation from end-to-end users. Note, also, that max-min fair proposals can harm FCT of short-lived flows as their sending rate is limited by the fair share.

Our goal is the design of a queue management algorithm that will integrate the design principles stated above: (1) max-min fairness of long-lived flows, and (2) small FCT of short-lived flows without use of explicit per-flow state[1] and without change of the existing TCP/IP infrastructure.

Our design is based on the observation that counting drops (and acting accordingly) provides enough information for achieving our goal. We propose a resource allocation algorithm, called Markov Active Yield (MAY), that satisfies both of our design principles yet is simple to implement, uses a single queue, scales for speeds of over $40Gbps$ and requires roughly one bit per flow of on-chip SRAM in Internet-like flow size distributions.

A. Paper contributions

Why is reaching the goal stated above hard? First, recall that in the max-min fair regime, a flow f experiences drops at *one and only one* link l_f at its path (we say that f is bottlenecked at l_f), and therefore must be protected at other links (that can be congested) by receiving lossless service. Second, if two or more flows are bottlenecked at the same link they must receive nonuniform loss rates that are function of their aggressiveness (round-trip times, TCP implementations, queuing delays, etc). Assuming that the router has access to the individual flow rates, or the existence of multiple queues that are appropriately scheduled, a number of solutions to these two problems exist, and are described in previous works [25], [11], [17], [5], [20]. However, in our case it is highly nontrivial to make the drop (mark) decision without any explicit information. On the other hand, existing router-based algorithms for minimizing of FCT, require either cooperation from the TCP sender [3], [10], or full state information [19] for *every* arriving flow, making them not scalable at very high speeds.

The main contributions of this paper are the following:

- A novel queue management scheme Markov Active Yield (MAY) that unifies two major design principles: small FCT of short-lived flows and max-min fair bandwidth sharing of long-lived flows. The *only* state information that is kept by MAY is a short history of *already dropped packets*.

- An analysis of the randomized algorithm MAY that shows that the bandwidth allocation of the long-lived elastic flows in the network of MAY queues is max-min fair. On the other hand, we show that nonelastic CBR flow receives (virtually) lossless service if its sending rate is less than max-min fair share, and asymptotic denial of service otherwise. Several representative packet level simulations are presented to verify our analytical findings.

- Analysis and evaluation of effects of MAY on FCT of short flows. MAY can reduce the FCT of TCP flows significantly compared to oblivious schemes (like RED) and instantaneously-fair packet schedulers (like DRR) and the FCT's are just slightly higher than those of LAS, a stateful scheme.

The fact that one can enforce max-min fairness by only counting past drops is not only theoretically surprising but is also of important practical interest. Namely, MAY memory requirements are roughly one bit (of on-chip SRAM[2]) per flow in Pareto-like flow size distributions (see Section III-D). Since computational complexity of the scheme is extremely low, MAY scales well at speeds greater than 40Gbps and several dozens of millions concurrent flows, with currently available

[1] Under explicit per-flow state we consider any method that can give an estimate of flow's arrival rate or number of previously arrived packets.

[2] Static RAM. Current implementations have access times of 2-6ns.

1-4244-1185-8/07/$25.00 © 2007 IEEE

hardware devices and state-of-the-art implementations of hash tables [21] and statistics counters [23].

B. Related work

Both of the problems that we study here have been studied extensively in the past. Here we classify existing proposals in two categories: (1) end-to-end transport protocols require change to cooperate with the routers; (2) end-to-end transport protocols do not require any change (routers are fully responsible for the resource allocation).

(1A) Bandwidth allocation of long-lived end-to-end users that cooperate with intermediate routers attracted a large attention in the last decade [22]. Examples include XCP [11], REM [2], MaxNet [26], etc.

(1B) Short FCT's of short-lived flows has been subject of interest of several recent studies. In [6], the authors propose an end-to-end protocol for reducing the FCT of short-lived flows. RuN2C [3] is a two-queues strategy, that cleverly exploits the 32-bit TCP-sequence-number space to reduce the FCT, by requiring a slight change in the TCP stack (Initial TCP-sequence-number must be set to an appropriate value in order to be prioritized by RuN2C).

(2A) In a similar manner to (1A), enforcing fairness "in the middle" without cooperation of end-to-end users has been a very active research area. Various algorithms exists: DRR [20], FQ [5], FRED [13], CSFQ [25], AFD [17], PD-RED [14] etc., all requiring explicit state information.

(2B) Size-aware scheduling policies like Shortest Job First (SJF), Shortest Remaining Processing Time (SRPT) or Least Attained Service (LAS) [12] have been used extensively in the area of operating systems to prioritize short jobs. However, in the Internet links, strategies that require knowledge of flow (job) sizes (such as SJF or SRPT) are not suitable, since that information is generally unavailable. On the other hand, stateful approaches have been shown to significantly improve FCT of short lived flows [19], [10].

II. Markov Active Yield (MAY)

In order to enforce fairness among long-lived users with different levels of aggressiveness, and responsiveness, it is clear that more aggressive and less responsive users must be punished more than others. The question is "by how much" and how this strategy should be implemented in an efficient manner. We would also like to have some kind of protection for "short-lived" flows. Here we call flow *short-lived* if it contains not more than S_0 packets.

Ideally, for a given S_0, all short-lived flows should be protected at a congested link, by receiving a lossless service. On the other hand, flows with more than S_0 packets should receive nonuniform loss rates that will result in fair bandwidth share. However, this idealized goal is hardly feasible (see [19]) at high speeds ($\geq 10Gbps$) since it would require keeping packet counters for each flow, strictly. The base for our discussion is the following powerful probabilistic paradigm that can reduce state information needed for more than two orders of magnitude, with minor performance degradation (in terms of FCT and max-min fairness):

```
1   OnPacketArrival(pkt, FlowID)
2     if FlowID ∈ H
3       with probability ν · δ(FlowID) do
4         drop(pkt);
5         ND(FlowID) + +;
6         TS(FlowID) = now;
7       enddo
8     else with probability q0 do
9       create(FlowID, H)
10      δ(FlowID) = 1;
11      ND(FlowID) = 1;
12      TS(FlowID) = now;
13      if CurrentUtilization > u0;
14        drop(pkt);
15      endif
16    enddo

17  Update(H)
18    if now − LastUpdate > Δ
19      ν = ν + κ(CurrentUtilization − u0);
20      for all FID in H
21        if now − TS(FID) > T0
22          remove(FID, H)
23        else
24          δ(FID) = (1 − qw)δ(FID) + qw ND(FID);
25          ND(FID) = 0;
26        endelse
27      endfor
28      LastUpdate = now;
29    endif
```

Fig. 1. Pseudocode of MAY.

(*) Drop proportionally to the amount of past drops

A queue management scheme that follows *(*)* is a feedback system: the more drops a flow experiences the higher will be the drop probability of that flow in the future. Sometimes, such feedback systems can diverge, but self-regulatory nature of TCP makes systems based on *(*)* stable (in terms of drop probabilities), as we shall see in Section III.

We will call flow f *new* if it has not experienced any loss until time t; otherwise we will say that f is old. A natural question that arises is what happens to new flows under strategy *(*)*. As we said, ideally we would like that all flows with size smaller than S_0, receive a lossless service and that all flows with size greater than S_0 join the "battle" for fair bandwidth share. Since we want to avoid the use of per-every-flow counters, we use a probabilistic argument to allow a few losses among short-lived flows and to enforce that flows with length greater than S_0 packets receive most losses in congested environments. To this end, we keep track of all old flows, and drop new-flow's packets with probability (of at most) $q_0 = \frac{1}{S_0}$[3]. This will be the input to the feedback system defined by *(*)*. As we will see in the next sections, a stateless strategy for dropping a new-flow's packets, enforces very few losses among flows with size less than S_0. On the other hand, easily implementable strategy *(*)* makes bandwidth allocations among arbitrary set of long-lived AIMD flows asymptotically independent of additive increase or multiplicative decrease parameters.

Now we proceed with a detailed description of MAY. As

[3]Rationale for this strategy is fact that expected number of prioritized packets of new flow is $\frac{1}{q_0} = S_0$.

u_0	desired utilization
Δ	length of update period
q_w	weighted average parameter
κ	controller gain
T_0	Timeout value

TABLE I
PARAMETERS OF MAY.

we have said the basic idea is to keep information of recently dropped packets and to use that statistics to regulate the drop probability for each of the flows. The complete pseudo-code is given in Figure 1.

The data structure used is a hash table H that stores quadruples $(FlowID, \delta(FlowID), TS(FlowID), ND(FlowID))$ for flows that have experienced some drops. Here $\delta(FlowID)$ is the drop frequency of the flow given by the fingerprint $FlowID$, $TS(FlowID)$ is the time stamp that tracks the time of last update of the given hash table entry and $ND(FlowID)$ is the number of dropped packets from the flow $FlowID$ during the current update interval of length Δ. At each packet arrival a packet's $FlowID$ is calculated. If there exists an entry in H that corresponds to $FlowID$, the arriving packet is dropped with probability proportional to the frequency of past drops: $\nu \cdot \delta(FlowID)$. The frequency of losses is calculated using weighted averaging (see line 24) with weight q_w. Finally the control variable ν determines the size of drop probabilities and therefore the utilization: if current utilization is less than desired (u_0) ν should be decreased to allow lower drop probabilities and increase utilization, while if current utilization is greater than u_0, then ν should be increased to allow higher drop probabilities and decrease utilization.

A. Implementation issues

State-of-the-art algorithms for high-speed hash table implementations presented in [21] allow implementations that run on line speeds greater than 40Gbps, assuming that hash table is stored in on-chip SRAM. In order to keep the whole hash table size small, we use the implementation of statistics counters architecture proposed in [23]. The size of MAY hash table entry can be made to be 10 bytes: 10 bits per counters $\delta(FlowID)$, $TS(FlowID)$ and $ND(FlowID)$); 30 bits for $FlowID$ fingerprint, and 20 bits for hash-table pointer. This would allow 16K hash-table entries on 1Mbit on-chip SRAM and 128K hash-table entries on 8Mbit SRAM. Probability of a hash collision is then $2^{-30}K^{-1}$ where K is the number of hash-table entries.

As we will se in the next Section, the memory requirements in Pareto-like flow-size distribution is around 1 bit per active flow. On the other hand, various publicly available OC192c and OC48c traces shows that the demand equivalent to *1Gbps* is generated by few hundred thousands of active flows (with timeout value of 64 seconds). Therefore, the memory requirements in the current internet flow-size distributions is roughly equivalent to one *Mbit* of on-chip SRAM per *Gbps* of line speed.

III. ANALYSIS OF MAY

A. Bandwidth allocation of long-lived elastic AIMD flows

In this subsection we prove that in steady-state an arbitrary set of AIMD flows [27] (flows with arbitrary linear increase parameter and arbitrary multiplicative decrease parameter), asymptotically obtain equal amount of throughput in a single bottleneck scenario (Corollary 3.1). Formally, throughout this subsection we assume the following.

Assumption 3.1: There are N long-lived flows $f_1, \ldots, f_N$ that use a congested link and all of them employ AIMD congestion control algorithms with an additive increase parameter $\gamma_i > 0$ and a multiplicative decrease $\beta_i \in (0,1)$, $i = 1, 2, \ldots, N$.

Under the previous assumption we can prove that MAY stabilizes ν and $\delta_i = \delta(f_i)$ (frequency of drops of flow f_i) in the model presented below.

Let $U_k^{(i)}$ be the throughput of flow f_i measured after k units of time Δ_0 (say a millisecond; We assume that $\Delta_0 \ll \Delta$). Denote the loss probability for a packet from flow (i) in time step k, between t-th and $t+1$-st hash-table update ($k\Delta_0 \in (t\Delta, (t+1)\Delta)$), by $\mu_i(k) = \delta_i(t)\nu(t)$. Having this, we can consider $U_k^{(i)}$ as a Markov chain on R^+ given by the transition:

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

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

From [7] we have that expected number of sent packets of flow i in time interval $(t\Delta, (t+1)\Delta)$ is

$$\bar{U}^{(i)}(t) = \frac{\beta_i}{1-\beta_i}\frac{\sqrt{\gamma_i}}{\sqrt{\delta_i(t)\nu(t)}} \cdot A \tag{1}$$

for a constant A, that does not depend on i (note that (1) is a generalized version of square root formula [16]). Denote by $\bar{\delta}_i(t)$ the number of dropped packets of flow f_i during time interval $(t\Delta, (t+1)\Delta)$. Each of $\bar{U}^{(i)}(t)$ packets is dropped with same probability $\delta_i(t)\nu(t)$, and we use the mean field approximation to estimate $\bar{\delta}_i(t)$:

$$\bar{\delta}_i(t) = \bar{U}^{(i)}(t)\delta_i(t)\nu(t) = \frac{\beta_i}{1-\beta_i}\frac{\sqrt{\gamma_i}}{\sqrt{\delta_i(t)\nu(t)}} \cdot A\delta_i(t)\nu(t) =$$

$$\frac{\beta_i}{1-\beta_i}\sqrt{\gamma_i}A\sqrt{\delta_i(t)\nu(t)} = a_i\sqrt{\delta_i(t)\nu(t)}, \tag{2}$$

where we denoted by $a_i = \frac{\beta_i}{1-\beta_i}\sqrt{\gamma_i}A$. On the other hand, we know that $\nu(t)$ is regulated to achieve utilization $u_0 C$, where C is the link capacity. Thus we have:

$$\sum_{i=1}^{N} \bar{U}^{(i)}(t) = \frac{1}{\sqrt{\nu(t)}}\sum_{i=i}^{N} a_i \frac{1}{\sqrt{\delta_i(t)}} = u_0 C. \tag{3}$$

From (2) and (3), we obtain

$$\bar{\delta}_i(t) = a_i\sqrt{\delta_i(t)}\sum_{i=1}^{N} a_i \frac{1}{\sqrt{\delta_i(t)}}\frac{1}{u_0 C}.$$

Since at the $t+1$-th update we use the weighted averaging: $\delta(t+1) = (1-q_w)\delta(t) + q_w\bar{\delta}_i(t)$, we conclude that the evolution of $\delta(t)$ is given by:

$$\delta_i(t+1) = (1-q_w)\delta_i(t) + q_w a_i\sqrt{\delta_i(t)}\sum_{j=1}^{N}\frac{a_j}{\sqrt{\delta_j(t)}}\frac{1}{u_0 C}. \quad (4)$$

Theorem 3.1: Suppose initially that $\delta_i(1) > 0$ for all $i = 1, \ldots, N$. Then in the mean field model, presented above, the system (3)-(4) is stable.

Before we proceed, note that by setting $z_i(t) = \delta_i(t)\frac{u_0 C}{a_i^2}$ equation (4) becomes equivalent to

$$z_i(t+1) = (1-q_w)z_i(t) + q_w\sqrt{z_i(t)}\sum_{j=1}^{N}\frac{1}{\sqrt{z_j(t)}}. \quad (5)$$

Without loss of generality, we can order z_i at time $t = 1$: $z_1(1) \le z_2(1) \le \ldots \le z_N(1)$. Directly from (5) we get that for all $t \ge 1$

$$z_1(t) \le z_2(t) \le \ldots \le z_N(t).$$

Denote by $r(t) = \frac{z_1(t)}{z_N(t)}$, the ratio between the smallest and the largest component of vector $z(t)$, and by $s(t) = \sum_{j=1}^{N}\frac{1}{\sqrt{z_j(t)}}$. Our first technical lemma shows that $r(t)$ is not decreasing function of t.

Lemma 3.1: Ratio $r(t) = \frac{z_1(t)}{z_N(t)}$ is nondecreasing function of t.

Proof: Since the order of z_i is preserved we have that

$$r(t+1) = \frac{z_1(t+1)}{z_N(t+1)} = \frac{(1-q_w)z_1(t) + q_w\sqrt{z_1(t)}s(t)}{(1-q_w)z_N(t) + q_w\sqrt{z_N(t)}s(t)}$$

$$= \frac{z_1(t)}{z_N(t)}\left(\frac{(1-q_w) + q_w\frac{1}{\sqrt{z_1(t)}}s(t)}{(1-q_w) + q_w\frac{1}{\sqrt{z_N(t)}}s(t)}\right) \ge \frac{z_1(t)}{z_N(t)} = r(t).$$

■

Lemma 3.2: The sequence $z_N(t)$ is bounded from above:

$$z_N(t) \le z_N(1) + \frac{N}{\sqrt{r(1)}} =: D.$$

Proof: First, note that $\sqrt{z_N(t)}\sum_{j=1}^{N}\frac{1}{\sqrt{z_j(t)}} < \sqrt{z_N(t)}\frac{N}{\sqrt{z_1(t)}} = \frac{N}{\sqrt{r(t)}} < \frac{N}{\sqrt{r(1)}}$. Now we prove the lemma by mathematical induction. For $t = 1$, statement is clearly true. Suppose that it is valid for $t = m$, then for $t = m+1$:

$$z_N(m+1) = (1-q_w)z_N(m+1) + \sqrt{z_N(m)}\sum_{j=1}^{N}\frac{1}{\sqrt{z_j(m)}} \le$$

$$(1-q_w)(z_N(1) + \frac{N}{\sqrt{r(1)}}) + q_w\frac{N}{\sqrt{r(1)}} < z_N(1) + \frac{N}{\sqrt{r(1)}} = D.$$

■

Having that $r(t) \le 1$ and that $r(t)$ is monotone, nondecreasing, we know that $r(t)$ converges to $r^* \le 1$. Next lemma shows that r^* is indeed 1.

Lemma 3.3: The sequence $r(t)$ converges to 1.

Proof: Suppose it is not true. Then there exist $\delta > 0$, such that for all t, $r(t) < 1 - \delta$. From the definition of $r(t)$, we have:

$$\frac{r(t+1)}{r(t)} = \left(1 + q_w\frac{(\frac{1}{\sqrt{z_1(t)}} - \frac{1}{\sqrt{z_N(t)}})s(t)}{1 - q_w + q_w\frac{1}{\sqrt{z_N(t)}}s(t)}\right) =$$

$$1 + \frac{q_w(\frac{1}{\sqrt{r(t)}} - 1)}{\frac{(1-q_w)\sqrt{z_N(t)}}{s(t)} + q_w} > 1 + \frac{q_w(\frac{1}{\sqrt{r(t)}} - 1)}{(1-q_w)z_N(t) + q_w} >$$

$$> 1 + \frac{q_w(\frac{1}{\sqrt{1-\delta}} - 1)}{(1-q_w)D + q_w} = E > 1.$$

Thus, assuming that $r(t) < 1 - \delta$, for all t, implies $r(t) > r(1)E^{t-1} \to \infty$, which is in turn a contradiction with $r(t) \le 1$. ■

Proof: (Theorem 3.1) Since $z_1(t)/z_N(t)$ converges to 1, we have that for every i, $\sqrt{z_i(t)}s(t)$ converges to N, and therefore, from (5) $z_i(t)$ converges to N. Now, from the definition of $z_i(t)$ we obtain that

$$\lim_{t\to\infty}\delta_i(t) = N\frac{a_i^2}{u_0 C} = \delta_i^*.$$

From (3), we conclude that

$$\lim_{t\to\infty}\nu(t) = \lim_{t\to\infty}\left(\frac{1}{u_0 C}\sum_{i=i}^{N} a_i\frac{1}{\sqrt{\delta_i(t)}}\right)^2 = \frac{N}{u_0 C} = \nu^*.$$

■

Corollary 3.1: In steady state, expected throughput of flow f_i does not depend on AIMD parameters, β_i and γ_i.

Proof: From (1),

$$\bar{U}^{(i)}(t) = \frac{\beta_i}{1-\beta_i}\frac{\gamma_i}{\sqrt{\delta_i^*\nu^*}}\cdot A = \frac{a_i}{\sqrt{N\frac{a_i^2}{u_0 C}\frac{N}{u_0 C}}} = \frac{u_0 C}{N}.$$

■

B. Bandwidth allocation of non-responsive CBR flows

In the previous subsection we have seen that set of N long-lived elastic AIMD flows bottlenecked at link with throughput $u_0 C$ with MAY queue receive, asymptotically, the same throughput: $u_0 C/N$. In this subsection we are going to explore the effects of MAY on the throughput of long-lived non-responsive CBR flows. Suppose that a link is shared by N elastic AIMD flows with AIMD parameters $\gamma_i > 0$ and $\beta_i \in (0,1)$ and M non-responsive CBR flows with constant sending rate of x_j packets per δ.

Theorem 3.2: In steady-state, the throughput of all AIMD flows is

$$U_i^* = \frac{1}{\nu^*}. \quad (6)$$

For the throughput of CBR flow T_j^* we have:

$$T_j^* = x_j \quad \text{if} \quad x_j \leq \frac{1}{\nu^*}, \tag{7}$$

$$T_j^* = 0 \quad \text{if} \quad x_j > \frac{1}{\nu^*}. \tag{8}$$

Here ν^* is the steady-state value of ν.

Proof: From the proof of the Theorem 3.1 we have (6). On the other hand denote by $\delta_j(t)$ weighted average of frequencies of drops from CBR flow c_j, and by $\bar{\delta}_j(t)$ the amount of drops from the same flow during the time step t. Then if $\delta_j(t_0)\nu^* \geq 1$ for some t_0, all packets starting after period t_0 will be dropped and $T_j^* = 0$. If $\delta_j(t)\nu^* < 1$

$$\delta_j(t+1) = (1-q_w)\delta_j(t) + q_w\bar{\delta}_j(t) =$$

$$= (1-q_w)\delta_j(t) + q_w x_j \delta_j(t)\nu^* = \delta_j(t)((1-q_w) + q_w x_j \nu^*).$$

Thus, for $x_j > \frac{1}{\nu^*}$, we have that $((1-q_w) + q_w x_j \nu^*) > 1$ and $\delta_j(t+1)$ is exponentially increasing until it becomes greater than $\frac{1}{\nu^*}$, and after that all packets from that flow are dropped. If $x_j < \frac{1}{\nu^*}$, $\delta_j(t+1) \to 0$ as $t \to \infty$, and therefore CBR flow g_j receives asymptotically lossless service, and $T_j^* = x_j$.

■

C. Max-min fairness of the elastic AIMD flows

The next theorem proves that, in steady-state, bandwidth allocations of elastic AIMD users is max-min fair.

Theorem 3.3: Let G be a network topology, with queues employing MAY and end users employing AIMD congestion control. For every flow r with steady-state rate x_r^* there exist a link on its path, such that the steady-state rates of all flows which traverse through that link are less then or equal to x_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 $R_{is} = 0$. By $\delta_i^{(s)*}$ denote the frequency of drops of flow i at link s, and by ν_s^* steady-state value of ν at link s. Then for arbitrary flow r there must exist link l for which $\delta_r^{(l)*} > 0$. On the other hand, from the definition of $\delta_r^{(s)*}$ we have that it satisfies the following equation.

$$\delta_r^{(s)*} = (1-q_w)\delta_r^{(s)*} + q_w x_r^* \nu_s^* \delta_r^{(s)*} = \delta_r^{(s)*}(1 - q_w + q_w x_r^* \nu_s^*). \tag{9}$$

Since we picked l such that $\delta_r^{(l)*} > 0$, from (9) we conclude that $1 - q_w + q_w x_r^* \nu_l^* = 1$, which implies $x_r^* = \frac{1}{\nu_l^*}$. Now, for any other flow r_1 that uses link l, $\delta_{r_1}^{(s)*} > 0$ implies $x_{r_1}^* = \frac{1}{\nu_l^*} = x_r^*$ while $\delta_{r_1}^{(s)*} = 0$ implies $x_{r_1}^* < \frac{1}{\nu_l^*}$ (see proof of Theorem 3.2). ■

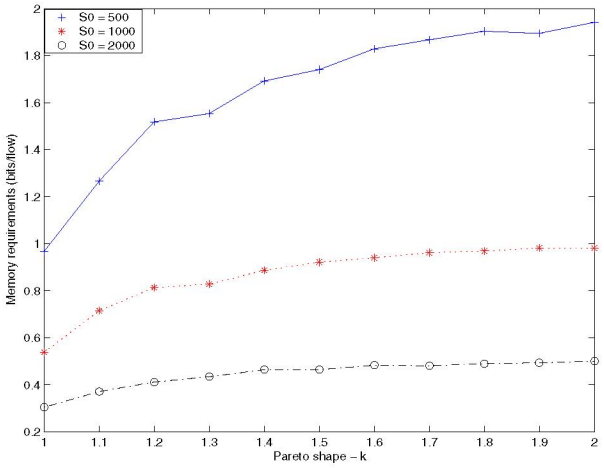

Fig. 2. Expected memory consumption (in bits per flow). Pareto shape $k \in [1, 2]$, mean $\mu = 10$, $500 \leq S_0 \leq 2000$.

D. Memory consumption

Let the sizes of flows be distributed according to the distribution $G(x)$:

$$P[|f| \geq x] = G(x) = \int_x^\infty g(y)dy.$$

Then the number of hash-table entries needed for a population of n flows: $(f_i)_{1 \leq i \leq k}$ is given by the random variable

$$Z(n) = \sum_{i=1}^{n} z(f_i),$$

where: $z(f_i) = 1$ if flow f_i has been hashed (experienced at least one loss), and $z(f_i) = 0$ if flow f_i has not been hashed (had no losses)

Denote by l_i the length of flow f_i. The probability that $z(f_i) = 0$ is:

$$P[z(f_i) = 0] = (1-q_0)^{l_i} = (1 - \frac{1}{S_0})^{l_i} \approx e^{-\frac{l_i}{S_0}}.$$

The expected memory consumption is then given by

$$E(Z(n)) = nE(z(f)) = n\int_0^\infty g(y)(1 - e^{-\frac{y}{S_0}})dy,$$

and the variance of $Z(k)$ is:

$$Var(Z(n)) = nVar(z(f)) = n\int_0^\infty g(y)(1 - e^{-\frac{y}{S_0}})e^{-\frac{y}{S_0}}dy.$$

The expected memory consumption (EMC) in bits per flow is given by $M(G) = 80E(Z(n))/n$ (we assume that each hash-table entry have $10bytes = 80bits$). Figure 2 contains EMC for Pareto distributions, with Pareto shape in $[1, 2]$ and mean 10 packets.

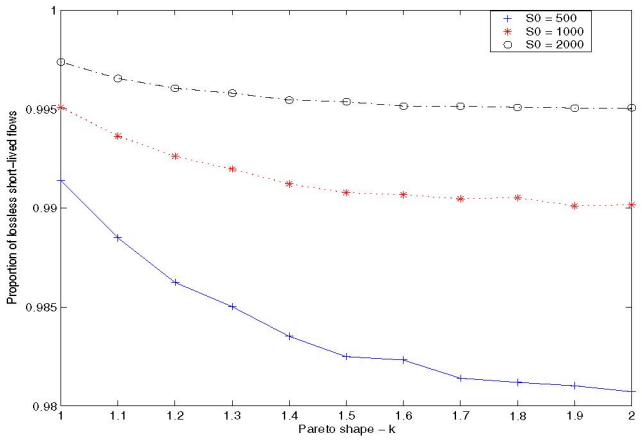

Fig. 3. Proportion of lossless short-lived flows. Pareto shape $k \in [1, 2]$, mean $\mu = 10$, $500 \leq S_0 \leq 10000$.

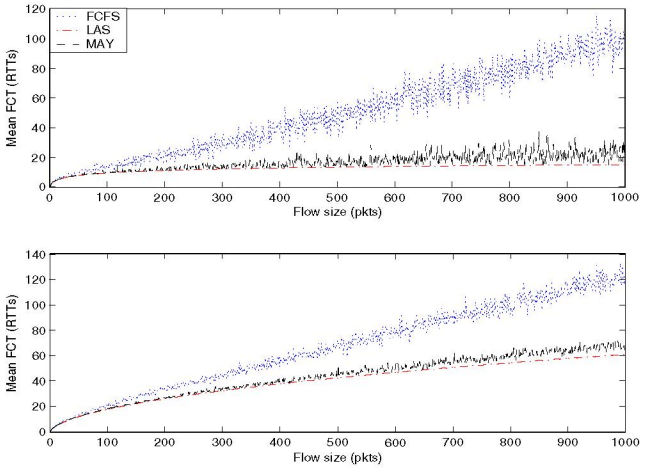

Fig. 4. Flow sizes versus FCT (measured in RTTs) in FIFO (with drop rate $p_0 = 0.01$), MAY ($S_0 = 1000$) and LAS cases. Top plot has no slow start limitations, while the bottom plot corresponds to $sstrhesh\_ = 2$.

E. FCT of short-lived TCP flows

A major factor that determines the FCT of a flow f is the congestion control algorithm (and its implementation) used by f. For example, in standard TCP, the parameters that directly determine the FCT are: the slow start threshold ($sstresh\_$), the advertised window, the maximum congestion window ($maxcwnd\_$), the delayed acknowledgements option, etc. In the simplest case (without $sstresh\_$ and $maxcwnd\_$ limitations) most short flows will complete quickly (in slow start phase), if they do not experience loss. In Pareto flow size environments, the proportion of short-lived flows that experience loss is very small. Namely, in population of k flows with Pareto distributed sizes, with shape $k \in [1, 2]$, and mean size μ proportion of short-lived flows that do not experience drops is

$$L(k, \mu, S_0) = Prob[z(f) = 0 | f < S_0].$$

For $\mu = 10$ and $k \in [1, 2]$ (standard values in the Internet) the value of $L(k, \mu, S_0) > 0.98$ for $S_0 > 500$ and $L(k, \mu, S_0) > 0.99$ for $S_0 > 1000$; see Figure 3.

To see how the FCT of TCP flows is affected by MAY one can consider a simple model of TCP with delayed acknowledgements (one ack per two packets) and without (upper) limitation in $cwnd\_$ where each packet is dropped with probability p. Figure 4 depicts the mean FCT (numerically obtained by averaging 100 runs) of short-lived TCP flows ($S_0 = 1000$) in three cases LAS ($p = 0$), MAY ($p = \frac{1}{S_0}$) and FIFO with drop rate $p = p_0 = 0.01$. Both $ssthresh\_ = 2$ and $ssthresh\_ = \infty$ are included. We can observe a slight increase in FCT between MAY and LAS, which is the consequence of the (stateless) probabilistic nature of MAY and price that must be paid for not keeping per-flow packet counters as in LAS. Figures are produced using the standard mathematical model of additive increase - multiplicative decrease $cwnd\_$ control and constant loss rate; see [16] or [7] for more details.

IV. Experimental results

In this section we briefly describe results of packet level $ns2$ simulations that demonstrate the behavior of MAY. We look at two issues:

- fairness of long-lived flows
- flow completion times of short-lived flows

To compare how far bandwidth allocations $U = (U_1, \ldots, U_N)$ deviate from max-min fair bandwidth allocations, $U_{mm} = (U_{1,mm}, \ldots, U_{N,mm})$, we use Jain's index the given by:

$$j(U) = \frac{\left(\sum_{i=1}^{N} \frac{U_i}{U_{i,mm}}\right)^2}{N \sum_{i=1}^{N} \left(\frac{U_i}{U_{i,mm}}\right)^2}. \quad (10)$$

Clearly, $j(U)$ has a global maximum 1 that is attained at $U = U_{mm}$ and since it is continuous, by measuring how far the index is from 1, one can get some intuition as to how far the vector U is from U_{mm}. The MAY parameters used in the experiments are: $\Delta = 1s$, $u_0 = 0.98$, $\kappa = 0.1$, $q_w = 0.05$, $T_0 = 64sec$. The DRR parameters are $No\_buckets\_ = 100$, $blimit\_ = 100Kbytes$, $quantum\_ = 1000bytes$. The self configuring Adaptive RED [9] is used to determine the RED parameters. TCP version is the standard TCP-SACK, with a packet size 1000B and delayed acknowledgements switched on. The aggressiveness of each flow is, thus, mainly determined by its RTT.

A. Fairness - Single bottleneck

The first set of simulations are designed to demonstrate the fairness properties of the proposed AQM schemes in single bottleneck scenario. Specifically, we present results for a single link with service rate of $80Mbps$ that services 100 long-lived TCP users with round trip times uniformly distributed in

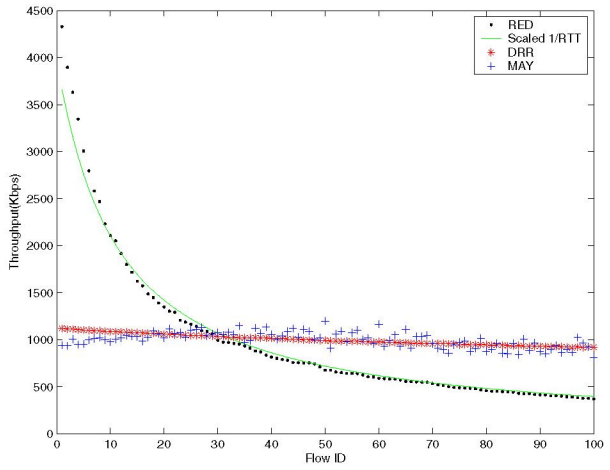

Fig. 5. Average throughput for 100 long-lived TCP flows over congested link employing RED, DRR and MAY.

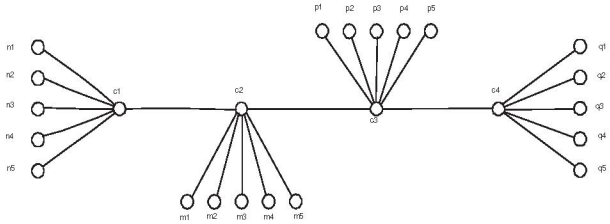

Fig. 6. Network topology

range $40-440ms$. To provide baseline results, we include the performance of RED and DDR for the same scenario. Share of the total throughput taken by each of 100 flows in these 3 schemes is depicted in Figure 5 during 2-minute simulations.

Jain's fairness indices for these three schemes are:

$$j(U_{RED}) = 0.606, \quad j(U_{DRR}) = 0.997, \quad j(U_{MAY}) = 0.993.$$

It can be seen from Figure 5 that the fairness of RED is approximately proportional to the inverse of RTT. This is in accordance with square root formula [16].

B. Fairness - Multiple bottleneck topology

Our second set of simulations demonstrate the fairness properties of MAY in network with multiple bottlenecks. The network topology that we considered is given in Figure 6. 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: $n(i) \rightarrow p(i); n(i) \rightarrow q(i), m(i) \rightarrow p(i); m(i) \rightarrow q(i); n(i) \rightarrow m(i); p(i) \rightarrow q(i)$ where $i = 1, 2, 3, 4, 5$.

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

$$ni \rightarrow c1 \ : \ 40 \cdot i + 1; \quad pi \rightarrow c3 \ : \ 40 \cdot i + 1$$
$$mi \rightarrow c2 \ : \ 40 \cdot i + 1; \quad qi \rightarrow c4 \ : \ 40 \cdot i + 1$$

and the delays $c1-c2$, $c2-c3$, $c3-c4$ are $10ms$. The capacities of all links are $10Mbps$. With this topology, the max-min fair

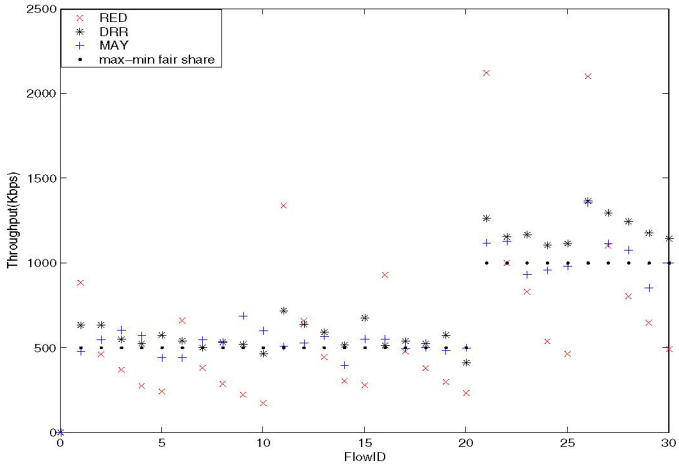

Fig. 7. Bandwidth obtained by each of 30 flows in multiple bottleneck topology. Congested links use RED, DRR, MAY.

shares are $0.5Mbps$ for 20 flows that uses link $c2-c3$, and $1Mbps$ for other 10 flows ($n(i) \rightarrow m(i)$ and $p(i) \rightarrow q(i)$).

The bandwidth allocations are evaluated with each link $c1-c2$, $c2-c3$ and $c3-c4$ using: RED, DRR and MAY, respectively, with a queue size of 100 packets.

Figure 7 depicts 3 scenarios, one for each dropping scheme used. We plot the amount of throughput taken by each of 30 flows during 2-minute simulations. The doted line represents the max-min fair share of bandwidth. We can see significant unfairness in oblivious (RED) scheme and close-to-max-min-fair resource allocation in DRR and MAY case. Jain's indices, defined by (10) are:

$$j(U_{RED}) = 0.731, \ j(U_{DRR}) = 0.987, \ j(U_{MAY}) = 0.985$$

C. Throughput of nonelastic flows

In this subsection we present simulations that support our analytical findings from the Theorem 3.2. The basic setup is the following: $N = 19$ TCP flows with RTTs uniformly distributed in $[20ms, 220ms]$ share a MAY link with capacity $40Mbps$ with a single CBR flow with sending rate x_1. We vary x_1 in the interval $[0.5Mbps, 8Mbps]$ and evaluated the throughput in each case. The results are presented in Figure 8. Each simulation lasted for 5 minutes, with first minute neglected. In this context, fair share is simply $x_f = 40/20Mbps = 2Mbps$. From Figure 8, we can see that for x_1 less than $x_f = 2Mbps$ throughout is identical to the sending rate, while for $x_1 > x_f$ CBR flow is shut down, by receiving (almost) zero throughput.

D. Flow sizes versus FCT of short-lived flows

In this subsection we present results that show how the FCT of short-lived flows (here we use $S_0 = 1000$) is impacted by the queue management algorithm used at a congested link. The basic setup is the following: 250 short-lived TCP flows share a $40Mbps$ bottleneck link with 50 long-lived TCP flows. Each connection has no $ssthresh\_$ nor $cwnd\_$ limitations,

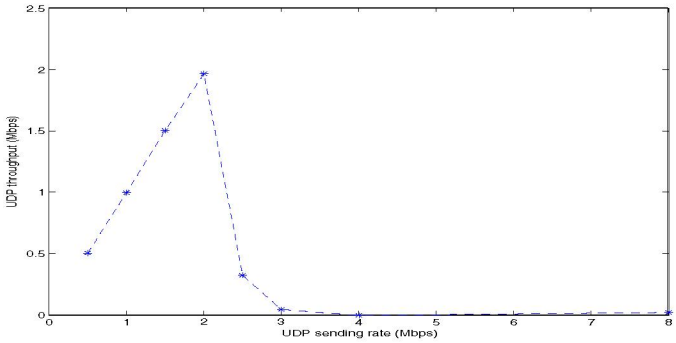

Fig. 8. Sending rate versus Throughput of the nonelastic UDP-CBR flow.

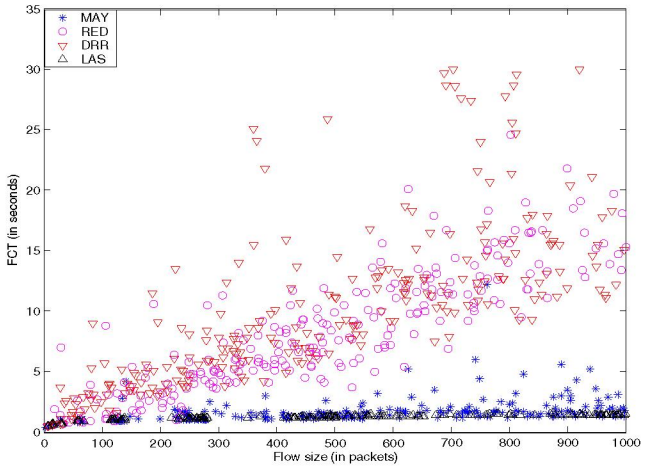

Fig. 9. Flow sizes versus FCT (measured in seconds) in RED, DRR, MAY and LAS cases.

delayed acknowledgements are switched on and packet sizes are $1000bytes$. RTT's of short-lived flows are $100ms$ (we use the same value in order to compare the FCTs of different flows, which are RTT-dependent), and RTT's of long-lived flows are uniformly distributed in $[20ms, 200ms]$. The sizes of short-lived flows are picked randomly with uniform distribution in the interval [1,1000] packets, and are initiated at rate of one per second ($\sim$ 10% of total load corresponds to short-lived flows). We evaluated the FCT of short lived flows obtained by four different queue management algorithms used by the bottleneck link: RED, DRR, LAS and MAY. The results are depicted in figure 9. Numerically, the average FCT (AFCT) for 250 short-lived flows in this four cases are:

$$AFCT_{RED} = 8.14s,\ AFCT_{DRR} = 10.34s,$$

$$AFCT_{LAS} = 1.26s,\ AFCT_{MAY} = 1.72s.$$

Note the slight increase of the FCT in MAY, compared to stateful scheme - LAS. The oblivious scheme (RED) and instantaneously fair packet scheduler (DRR) achieve significantly larger FCT. In RED no packet is prioritized while in DRR the instantaneous sending rate is limited and most of short-lived flows experience loss during the slow-start phase.

V. Discussion of MAY

Estimating the fair share and the number of bottlenecked flows. From the analysis in section III we know that max-min fair share x^* can be estimated with

$$x^* = \frac{1}{\nu^*}.$$

From the proof of Theorem 3.3, we also know that a flow f is bottlenecked at the link l if and only if δ_f^* is asymptotically positive. In practice, we can estimate the number of bottlenecked flows at link l at time t as

$$NB(l) = \#\{f \in H : \delta_f(t) > \epsilon\}$$

for some $\epsilon > 0$.

Isolating the high-rate nonelastic flow. Nonelastic flows that have sending rate higher than responsive flows bottlenecked at given link would get asymptotically full denial of service by MAY. One may argue that this is not desirable feature of MAY, but since the nonelastic flows (with high sending rate) have few orders of magnitude higher drop rates compared to elastic flows they can be easily identified and can receive additional processing by the router if required.

Variable packet size. Throughout this paper we assumed uniform packet sizes (to ease the exposition). The problem of variable packet size can be easily solved by employing byte-based approach instead packet-based approach.

Control of ν. In the present implementation the control of ν is rate based, rather than queue-based. This means that congestion indicator is arrival rate at the queue, rather than queue length. Queue based control of ν is possible as well without affecting the main features of MAY.

Definition of short-lived flow. In our terminology a flow is short-lived if it has length not greater than S_0 packets. The quantity S_0 determines memory consumption as well as the amount of traffic prioritized. S_0 should be chosen to allow small FCT of majority of flows, but not to harm performance of long-lived flows. Our measurements over real Internet traces indicate, for example, that $S_0 = 1000$ implies roughly 10-20% of traffic generated by short-lived flows. The Figure 10 depicts the proportion of short-lived-flow traffic as function of S_0 at two real internet traces [15]: Leipzig and Abilene. We can see, for example, that picking $S_0 = 1000$ corresponds to 10% of trafic in the Abilene trace and 24% in the Leipzig trace. We also note that it is possible to self-tune the parameter S_0 to allow certain, prescribed, proportion of traffic to be prioritized.

Parameter calibration. Initial tests show that MAY is highly robust to the choice of parameters. The update interval Δ should be taken to cover several "typical" RTT values, thus to belong to interval $[500, 5000]ms$. The weighted average q_w should be chosen to allow averaging over several update intervals: $q_w \in [0.01, 0.1]$. Timeout T_0 depends on the definition of persistence of a flow. Standard value used in Internet measurements is $T_0 = 64sec$, and here we use the same value. Self tuning of parameters is possible but is out of scope of the present paper.

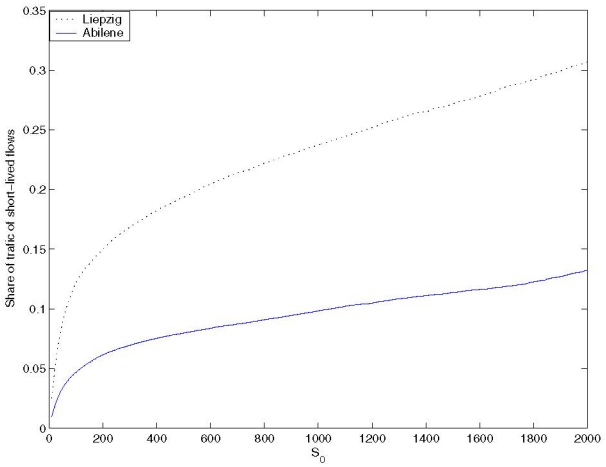

Fig. 10. Share of traffic produced by short-lived flows (those with less than S_0 packets) as function of S_0. Real Internet traces: Leipzig and Abilene.

VI. Summary

In this paper we proposed a randomized framework for unifying two major principles for the design of resource allocation algorithms at Internet links: small Flow Completion Time of short-lived flows and max-min fair bandwidth sharing of long-lived flows. While both problems have been extensively studied, over last two decades, no stateless solution exist for either of them. A powerful paradigm "Drop proportionally to the amount of past drops", allows us to develop a virtually stateless scheme, called MAY which unifies two design principles using around 1 bit per flow of on-chip SRAM in Internet-like flow size distributions. The light computational complexity and recent advances in hash-table implementations allow implementation at line speeds greater than 40Gbps.

Our analytical work, shows that for elastic AIMD flows, the amount of bandwidth obtained by an elastic long-lived flow asymptotically does not depend on its aggressiveness (additive increase factor) neither responsiveness (multiplicative decrease factor). We also show that in MAY loss rates of nonelastic, CBR flows, exhibit phase transition: if sending rate of a CBR flow f_0 is not greater than max-min fair share the loss rate of f_0 is zero otherwise the loss rate of f_0 is 1. The fairness level of MAY, evaluated using Jain's fairness index, is almost identical to one of DRR. On the other hand, increase of FCT for short-lived flows compared to stateful LAS is minimal and is consequence of probabilistic nature of our scheme, and price paid for avoidance of explicit state information.

Finally, we note that although the analysis in this paper has been concerned with classes of AIMD and CBR flows, it can be extended to any other transport protocol Λ for which the throughput $U_\Lambda(p)$ is a decreasing function of loss rate p, which is a principle feature of any congestion aware TCP version[4].

[4]See, for example, [1] for an analysis of $U_\Lambda(p)$ under MIMD congestion control.

VII. Acknowledgements

This work is supported by the Science Foundation Ireland grant 04/IN3/I460. The authors would like to thank Dave Malone for proofreading of an early draft of this paper and the anonymous reviewers for useful comments and suggestions.

References

[1] E. Altman, K. Avrachenkov, C. Barakat, A. Alam Kherani, B. J. Prabhu. "Analysis of MIMD congestion control algorithm for high speed networks". Computer Networks 48(6): 972-989 (2005)

[2] S. Athuraliya, D. Lapsley, S. Low. "Enhanced Random Early Marking algorithm for Internet flow control". Proc. IEEE INFOCOM, 2000.

[3] K. Avrachenkov, U. Ayesta, P. Brown, E. Nyberg. "Differentiation between short and long TCP flows: Predictability of the response time". In Proceedings of IEEE INFOCOM, Hong Kong, 2004.

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

[5] A. Demers, S. Keshav, S. Shenker. "Analysis and simulation of a fair queueing algorithm". Proc. ACM SIGCOMM, Austin, TX, USA, 1989.

[6] N. Dukkipati, M Kobayashi, R. Zhang-Shen, N. McKeown. "Processor Sharing Flows in the Internet". In Proceedings of IWQoS, Passau, 2005.

[7] V. Dumas, F. Guillemin, P. Robert. "A Markovian analysis of additive-increase multiplicative-decrease algorithms". Adv. in Appl. Probab. 34 (2002), no. 1, 85-111.

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

[9] S. Floyd, R. Gummadi, S. Shenker. "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management". August 2001, online: http://www.icir.org/floyd/papers/adaptiveRed.pdf.

[10] L. Guo and I. Matta. "The War between Mice and Elephants". In Proceedings of IEEE ICNP, USA, 2001.

[11] D. Katabi, M. Handley, C. Rohr. "Internet Congestion Control for Future High Bandwidth-Delay Product Environments". In Proceedings of ACM SIGCOMM, Pittsburgh, PA, USA, 2002.

[12] L. Kleinrock. "Queuing Systems Vol. II: Comp. App.". Wiley, 1976.

[13] D. Lin, R. Morris. "Dynamics of random early detection". In Proceedings of ACM SIGCOMM, Cannes, France, 1997.

[14] R. Mahajan, S. Floyd, D. Wetherall. "Controlling high-bandwidth flows at the congested router". In Proceedings of IEEE ICNP, CA, USA, 2001.

[15] NLANR IP traces, online: http://pma.nlanr.net/.

[16] J. Padhye, V. Firoiu, D. F. Towsley, J. F. Kurose. "Modeling TCP Reno performance: a simple model and its empirical validation". *IEEE/ACM Transactions on Networking*, Volume 8 , Issue 2 ,April 2000.

[17] R. Pan, L. Breslau, B. Prabhakar, S. Shenker. "Approximate Fairness through Differential Dropping". ACM SIGCOMM Computer Communication Review Volume 33 , Issue 2, April 2003

[18] R. Pan, B. Prabhakar, K. Psounis. "CHOKe: A stateless AQM scheme for approximating fair bandwidth allocation". In Proceedings of IEEE INFOCOM, Tel Aviv, Israel, 2000.

[19] I.A. Rai, E.W. Biersack, G. Urvoy-Keller. "Size-based scheduling to improve the performance of short TCP flows". IEEE Network, 2005.

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

[21] H. Song, S. Dharmapurikar, J. Turner, J. Lockwood. "Fast hash table lookup using extended bloom filter: an aid to network processing". In Proceedings of SIGCOMM, Philadelphia, USA, 2005.

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

[23] R. Stanojevic. "Small active counters". In Proceedings of IEEE Infocom 2007, Anchorage, USA.

[24] R. Stanojević, R. Shorten. "Beyond CHOKe: Stateless fair queueing". Proceedings of NET-COOP 2007, LNCS, Springer, vol.4465.

[25] I. Stoica, S. Shenker, H. Zhang. "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.

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

[27] Y. Yang, S. Lam. "General AIMD Congestion Control". In Proceedings of ICNP 2000, Osaka, Japan, 2000.