Improving TCP Fairness with the MarkMax Policy

Natalia Osipova, Alberto Blanc, and Konstantin Avrachenkov,
INRIA Sophia Antipolis,
2004, route des Lucioles - B.P.93 - 06902 Sophia Antipolis, France
{Natalia.Osipova, Alberto.blanc, K.Avrachenkov}@sophia.inria.fr

Abstract

We introduce MarkMax a new flow-aware Active Queue Management algorithm for Additive Increase Multiplicative Decreases protocols (like TCP). MarkMax sends a congestion signal to a selected connection whenever the total backlog reaches a given threshold. The selection mechanism is based on the state of large flows. Using a fluid model we derive some bounds that can be used to analyze the behavior of MarkMax and we compute the per-flow backlog. We conclude the paper with simulation results, using NS-2, comparing MarkMax with Drop Tail and showing how MarkMax improves both the fairness and link utilization when connections have significantly different Round Trip Times.

Keywords: AQM, TCP, Drop Tail, MarkMax, maxmin fairness

1 Introduction

It has been known for a long time that if TCP (Transmission Control Protocol) connections with different RTTs (Round Trip Times) share a bottleneck link, TCP connections with smaller RTTs take a larger share of the bandwidth [10, 13]. In [12] the authors have observed that under synchronization assumptions a TCP connection obtains a share of the link capacity proportional to RTT^{α} with $1 < \alpha < 2$. In [9] the author has used a fluid approximation to derive a more rigorous model for the case when connections have different RTT. Then, in [3] it was observed that in the case of not complete synchronization and, especially when RED [11] is used, the distribution of the link capacity is more fair. In particular, the experiments of [3] have suggested that a TCP connection obtains a share of the link capacity proportional to $RTT^{0.85}$. This was later justified by an analytical model for the case of two competing TCP connections [5]. In [2] the authors have used a fluid model to analyze what happens if only one connection reduce its sending rate when multiple connections share the same bottleneck link but they have ignored backlog dynamics: whenever the total arrival rate at the bottleneck link is equal to its capacity one of the connection reduces its sending rate, so that the backlog is always zero. In [18] the authors have proposed an MLC(l) AQM (Active Queue Management) algorithm to approach maxmin fairness. In particular, for $l = 1$ the MLC(l) algorithm performs similar to RED and for $l = 2$ the MLC(l) algorithm performs similar to CHOKe [14]. The authors of [18] argue that by choosing a significantly large parameter l one can be arbitrary close to the maxmin fairness. The present work indicates that this does not appear to be the case.

Building upon [8, 2] and [17] we propose a new *flow-aware* active queue management packet dropping scheme (MarkMax). The main idea behind MarkMax is to identify *which connection should reduce its sending rate* instead of *which packets should be dropped.* To improve fairness we propose to cut flows with the largest sending rate during the congestion moments. Several AQM schemes previously proposed do not discriminate between flows. Typically they drop every incoming packet with a certain probability that is a function of the state of the queue.

When AQM was first introduced in the 1990s it was unfeasible to classify incoming packets in real time for high speed links but with technological advances this is now possible. Furthermore, to reduce the numbers of flows that need to be tracked, it is possible to concentrate on the larger flows using the heavy-hitter counters of [17] to identify large flows. Then, according to [6] we suggest to treat short flows with priority and mark large flows which have the largest backlog during the congestion moments. We also suggest to use Explicit Congestion Notification (ECN) [15] to minimize the number of dropped packets.

The paper is organized as follows: In the next Section 2 we specify the algorithm. Then, in Section 3 we perform its theoretical analysis. We conclude the paper with a section on NS-2 simulations illustrating the performance of MarkMax.

2 The MarkMax Algorithm

The algorithm has three parameters: the thresholds θ, θ_l, θ_h, selected in such a way that $\theta_l < \theta < \theta_h$. The threshold θ acts as a "trigger," whenever the queue size is above this value one connection is cut. We propose two different ways of selecting which connection to cut, as described later on. The other two thresholds are needed because we are dealing with a packet based system with non-zero propagation and queueing delays.

Let q be the queue size and *flag* be a Boolean variable initialized to TRUE. The following algorithm is executed every time a new packet arrives:

```
enqueue packet
if $q \leq \theta_l$ or $q \geq \theta_h$
    then $flag \leftarrow$ TRUE
if $q \geq \theta$ and $flag$=TRUE
    then a. select connection with MarkMax-B (full
              backlog based MarkMax) or MarkMax-T
              (backlog tail based MarkMax)
         b. set the ECN flag in the first packet of the se-
              lected connection from the head of the queue
         c. $flag \leftarrow$ FALSE
```

The θ_h and θ_l thresholds are used do determine whether a congestion signal should be sent or not, if $q \geq \theta$. After a congestion signal is sent the algorithm will not send another one as long as the queue remains in the interval $[\theta - \theta_l, \theta + \theta_h]$.

The θ_h threshold acts as a safety mechanism covering the cases when a single cut in the sending rate of the selected connection might not be enough to reduce the *total* arrival rate to a value smaller than the capacity of the outgoing link. Whenever the queue size is above θ_h we keep sending congestion signals to the selected connection. This does happen especially during the slow start phase.

Given that the system has non-zero propagation and queueing delays whenever we set the ECN bit of a certain connection we need to wait for the sender to receive the corresponding acknowledgment before it reduces its sending rate. Before such reduction is noticeable at the bottleneck link we still need to wait for the propagation and queueing delay between the sender and the bottleneck link. During this time the sending rate and the queue will keep growing so that, at the bottleneck link, it is not immediately possible to conclude whether a single cut is enough or not. Clearly if we set θ_h too high the system will respond slowly, whenever one cut is not enough, and the queue will be larger. On the other hand if we set θ_h too close to θ unnecessary multiple cuts can take place.

The lower threshold θ_l is needed because the queue size can oscillate around θ, due to the arrival and departure of single packets and to the bursty nature of the arrival flows. If the queue size is close to θ the threshold can be crossed multiple times, so if we use only one threshold θ this could generate multiple congestion signals, potentially causing the sender to reduce its sending rate multiple times[1]. Furthermore it could happen that different connections are selected, causing, again, multiple and unnecessary cuts. Because of these oscillations using θ_l is the only way to determine whether the selected connection has reacted to the congestion signal.

Even if a single cut is enough to reduce the total sending rate to a value smaller than the capacity of the outgoing link the additive increase aspect of TCP will increase the sending rate again so that the backlog will, eventually, start to increase again. Clearly if we set θ_l too low the backlog might never reach it forcing the algorithm to use only the θ_h threshold and to send multiple spurious congestion signals.

In the next section we use a fluid approximation to further discuss the selection of θ and θ_h. Based on the simulations we run it looks reasonable to suggest that θ_h and θ_l can be set as follows: $\theta_h = 1.15 \cdot \theta$ and $\theta_l = 0.85 \cdot \theta$.

After enqueueing the arriving packet the algorithm sets the *flag* variable to TRUE if the queue size has grown too large or has sufficiently decreased. In both cases the queue size is sufficiently far from θ so that we should send a new congestion signal if $q \geq \theta$. This is done by the last **if** statement: at first a connection is selected, then the ECN flag is set in the first packet from the head of the queue of the selected connection. Finally the *flag* is set to FALSE to indicate the fact that one congestion signal has already been sent.

We propose two different criteria for selecting the connection to be cut: MarkMax-B selects the connection with the biggest (per connection) backlog and MarkMax-T selects the connection with the biggest backlog in the final part of the queue (the tail). As such the MarkMax-T variant has one extra parameter, expressed as a percentage, indicating the portion of the queue that will be considered.

The per connection backlog is related to the sending rate of each connection. Clearly a larger sending rate will result in larger backlog. More precisely the connection with the biggest backlog is the connection with the largest *average* sending rate since the beginning of the current busy period. Larger values of θ and corresponding larger queues lead to a larger averaging window, basically increasing the "memory" of the system. The idea behind MarkMax-T is to reduce the averaging window in order to identify the connection with the biggest *instantaneous* rate.

[1]The ECN specification does mention that senders should reduce the sending rate only once per round trip time, but this is not enough to guarantee that multiple cuts will not take place if we mark multiple packets.

3 Fluid Model

Consider N TCP connections sharing a single bottleneck link with capacity μ. Let RTT_i be the round trip time of the i-th connection ($i = 1, \ldots, N$) and $\lambda_i(t)$ its sending rate at time t. We approximate the behavior of the system using a fluid model. Data is represented by a fluid that flows into the buffer with rate $\lambda(t) = \sum_i \lambda_i(t)$, and it leaves the buffer with rate μ if there is a non-zero backlog. Fluid models have been successfully used to model TCP connections. In [1] it is shown that such a model adequately describes the behavior of a TCP connection, provided the average sending rate is large enough.

As in [7] we assume that, between congestion signals, senders increase their sending rate linearly. If at time t_0 the sending rate of the i-th sender is $\lambda_{0,i}$ then at time $t > t_0$ its sending rate is $\lambda_i(t) = \lambda_{0,i} + \alpha_i t$, where $\alpha_i = 1/(RTT_i)^2$. For the sake of simplicity we assume that RTT_i is a constant, as if it often done (see, for example, [4, 16, 7]).

It is not too hard to see that, if at time t_0 the sending rates are $\lambda_{0,i}$ and the total backlog is x_0, the backlog $x(t)$ is given by:

$$x(t) = x_0 + (\lambda_0 - \mu)t + \frac{\alpha}{2}t^2. \tag{1}$$

Where $\lambda_0 = \sum_i \lambda_{0,i}$ and $\alpha = \sum_i \alpha_i$, provided x_0 and λ_0 are such that $x_0 \geq \frac{(\lambda_0 - \mu)^2}{2\alpha}$. If $x_0 < \frac{(\lambda_0-\mu)^2}{2\alpha}$ and $\lambda_0 < \mu$ then, after a decreasing phase, the buffer will be empty for a certain time and will finally start increasing again. In this case

$$x(t) = \begin{cases} x_0 + (\lambda_0 - \mu)t + \frac{\alpha t^2}{2}, & \text{if } t \leq t_1 \\ 0, & \text{if } t_1 \leq t \leq \frac{\mu - \lambda_0}{\alpha} \\ \frac{\alpha}{2}\left(t - \frac{\mu-\lambda_0}{\alpha}\right)^2, & \text{if } t > \frac{\mu-\lambda_0}{\alpha} \end{cases} \tag{2}$$

where $t_1 = \frac{\mu - \lambda_0 - \sqrt{(\mu-\lambda_0)^2 - 2\alpha x_0}}{\alpha}$.

Solving $\lambda(t) = \lambda_0 + \alpha t$ for t and substituting in (1) we have that

$$x(\lambda) = \frac{\lambda^2}{2\alpha} - \frac{\lambda\mu}{\alpha} + x_0 + \frac{\mu\lambda_0}{\alpha} - \frac{\lambda_0^2}{2\alpha} \tag{3}$$

provided $x_0 \geq \frac{(\lambda_0-\mu)^2}{2\alpha}$. A similar expression can be obtain substituting the value of t in (2). Figure 1 shows some of the possible trajectories of the system. Note that all these parabolas have the same shape in the sens that as x_0 and λ_0 vary the only thing that changes is the height of the vertex on the $\lambda = \mu$ line.

One possible way of adapting the MarkMax algorithm to the fluid case is as follows: every time the total backlog $x(t)$ reaches θ we can "send a congestion signal" to the corresponding connection by multiplying its sending rate by β ($0 < \beta < 1$). Throughout the paper we will use $\beta = 1/2$ (to model TCP New Reno) unless otherwise stated. The two selection methods previously discussed can easily be adapted as well: for MarkMax-B we select the connection with the biggest backlog, while for MarkMax-T we pick the connection with the biggest instantaneous sending rate. Recall that the idea behind MarkMax-T was exactly this and, with the fluid model, we know $\lambda_i(t)$ exactly so there is no need to approximate it.

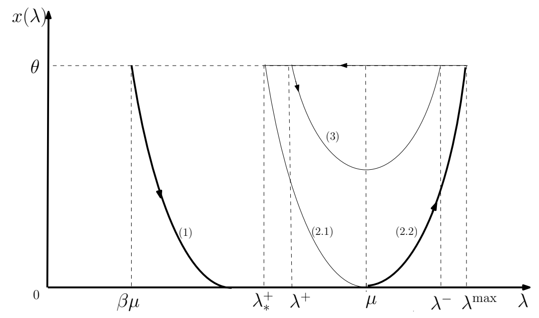


Figure 1: Some of the possible trajectories in the state space

To simplify the analysis, unless otherwise specified, we assume that the source reacts immediately to the congestion signals. Combining this with the fact that we know the sending rate after a cut and the are no short term oscillations in the queue size, it suffices to use only one threshold (θ). As a consequence whenever the backlog reaches θ the chosen connection, say j, immediately changes its rate to $\beta\lambda_j$. If $\sum_{i \neq j} \lambda_i + \beta\lambda_j > \mu$ (that is the arrival rate is still greater than μ) the procedure is repeated by selecting a new connection to cut (it can be the same one or not, depending on the specific case) until the total sending rate is less than μ. For the MarkMax-T version this procedure is guaranteed to terminate: eventually all connections will be cut. While for MarkMax-B this is not the case: if multiple cuts are needed the algorithm will always pick the same connection. As there is no feedback delay the backlog does not change. If the sum of the rates of the other connections is greater than μ even an infinite number of cuts will not suffice and the algorithm will not terminate. Given that this happens only in the fluid model and only for very large (and unrealistic) values of θ we decided not to address the problem.

It is worth noting that using the fluid model we just presented it is also possible to exactly compute the *per connection* backlog, at any given time t, using an approach based on *network calculus*. Let $R_{i,\text{in}}(t)$ be the *total* amount of traffic sent by the i-th connection until time t (this is generally called a "process" in network calculus), that is $R_{i,\text{in}}(t) = \int_0^t \lambda_i(u)du$. Similarly let $R_{i,\text{out}}(t)$ be the total amount of traffic of connection i that has left the buffer until time t. Clearly the backlog at time t is $x_i(t) = R_{i,\text{in}}(t) - R_{i,\text{out}}(t)$ so that we need to compute $R_{i,\text{in}}(t)$ and $R_{i,\text{out}}(t)$ to find $x_i(t)$. Let $t_1, \ldots, t_n$ be the times at which a congestion signal was sent (to any of the connections). Between two congestion signals, say t_j and t_{j+1}, we know that if $\lambda_i(t) = \lambda_{i,j} + \alpha_i t$ then $R_{i,\text{in}}(t) = S_{i,j} + \lambda_{i,j}(t - t_j) + \frac{\alpha_i}{2}(t - t_j)^2$ where $\lambda_{i,j} \triangleq \lambda_i(t_j)$ and $S_{i,j} \triangleq R_{i,\text{in}}(t_j)$. This way we can also compute $R_{i,\text{in}}(\tau)$ for any $\tau \leq t$.

To compute $R_{i,\text{out}}(t)$ we can take advantage of the fact that we are dealing with a fluid FIFO queue with continuous inputs (the arrival rate is bounded) so that the delay for all the bits exiting at time t is the same and it is equal $d(t) = \inf\{u \geq 0 | R_{\text{in}}(t - u) \leq R_{\text{out}}(t)\}$ where $R_{\text{in}}(t) = \sum_i R_{i,\text{in}}(t)$ and $R_{\text{out}}(t) = \sum_i R_{i,\text{out}}(t)$. This implies that $R_{i,\text{out}}(t) = R_{i,\text{in}}(t - d(t))$. As we know $R_{i,\text{in}}(\tau)$ for any $\tau \leq t$ we only need to find $d(t)$ to compute $R_{i,\text{out}}(t)$.

Let $v \triangleq t - d(t)$, that is the bits that are exiting at time t joined the queue at time v. We can find v exploiting the fact that $R_{i,\text{in}}(v) = R_{\text{out}}(t)$ and that $R_{\text{out}}(t) = \mu(t - u)$, where u is the beginning of the system busy period containing t and can be found because $R_{i,\text{in}}(\tau)$ is known for all $\tau \leq t$. We also have that, if $t_k \leq \tau \leq t_{k+1}$:

$$R_{\text{in}}(\tau) = S_k + \lambda_0(\tau - t_k)\frac{\alpha}{2}(\tau - t_k)^2, \tag{4}$$

where $S \triangleq R_{\text{in}}(t)$, $k \triangleq \max\{j | S_j < S\}$ and $S_j \triangleq \sum_i S_{i,j}$. That is the traffic exiting at time t entered in the buffer between t_k and t_{k+1}. As $t_k \leq v \leq t_{K+1}$ we can use (4) to solve $R_{\text{in}}(v) = \mu(t - u)$ for v and finally compute $d(t) = t - v$. Knowing $d(t)$ we can use $R_{i,\text{out}}(t) = R_{i,\text{in}}(t - d(t))$ to compute $x_i(t) = R_{i,\text{in}}(t) - R_{i,\text{out}}(t)$.

Using this method we wrote a simulator for the fluid model (in Python) that implements both variants of MarkMax. Using this simulator we have noticed that, provided the value of θ is not too big, MarkMax-B and MarkMax-T behave in a very similar way. In the remainder of this section we present some results that can be derived using the fluid model.

3.1 Guideline Bounds

Let t_θ be such that $x(t_\theta) = \theta$. Let λ^- and λ^+ be, respectively, the total sending rate before and after the cut(s) at time t_θ. Let

$$g(\lambda) = \begin{cases} 0, & \text{if } \lambda \leq \mu \\ \frac{(\lambda-\mu)^2}{2\alpha}, & \text{if } \lambda \geq \mu \end{cases},$$

(marked as (2.2) in Figure 1) and let $A = \{(\lambda, x) | x > g(\lambda)\}$. It is easy to verify that if $(\lambda_0, x_0) \in A$, then any trajectory starting at (λ_0, x_0) stays in A. Furthermore, given that we send the congestion signal(s) whenever $x(t) = \theta$ and that there is no feedback delay, the maximum rate $\lambda^{\max}$ corresponds to intersection between $g(\lambda)$ and $x = \theta$ in Figure 1. It is easy to see that $\lambda^{\max} = \mu + \sqrt{2\alpha\theta}$.

Clearly all the trajectories described by (3) intersect the $x = \theta$ line twice, once to the left of the $\lambda = \mu$ line and once to the right. Only the intersection points to the right correspond to an increasing backlog phase so that λ^- is always between μ and $\lambda^{\max}$. We can also bound λ^+: as we keep sending congestion signals until the arrival rate is less than μ we have $\lambda^+ \leq \mu$. The fact that $\lambda^- \geq \mu$ implies that λ^+ cannot be smaller than $\beta\mu$ (this happens when $\lambda^+ = \mu$ and either there is only one connection or, in the case of multiple connections, the biggest one is significantly bigger than the others). Combining all this we have:

$$\mu \leq \lambda^- \leq \mu + \sqrt{2\alpha\theta}, \tag{5}$$
$$\beta\mu \leq \lambda^+ \leq \mu. \tag{6}$$

After the cut(s) the total sending rate will be reduced by a factor $\tilde{\beta} \triangleq \lambda^+/\lambda^-$, which is always smaller than β.

Lemma 1. *If we use MarkMax-T then:*

$$\frac{\beta}{1 + \sqrt{2\alpha\theta}/\mu} \leq \tilde{\beta} \leq \frac{N + \beta - 1}{N}. \tag{7}$$

Proof. Let λ_i^- be the sending rate of i-th connection at time t_θ so that $\lambda^- = \sum_i \lambda_i^-$. And let j be such that $\lambda_j = \max_i\{\lambda_i\}$, then:

$$\begin{aligned}\tilde{\beta} &= \frac{\lambda^+}{\lambda^-} \leq 1 - (1 - \beta)\frac{\lambda_j^-}{\lambda^-} \\ &\leq 1 - (1 - \beta)\frac{\lambda^-}{\lambda^- N} = \frac{N - 1 + \beta}{N}.\end{aligned}$$

Where the first equality is the definition of $\tilde{\beta}$, the first inequality follows from the fact that $\lambda^+ \leq \beta\lambda_j^- + \sum_{i \neq j} \lambda_j^- = \lambda^- - \lambda_j^-(1 - \beta)$; this inequality is true because the right hand side corresponds to the case where there is only a single cut and in this case λ^+ is largest. The second the inequality follows from the fact that $\lambda_j = \max_i\{\lambda_i\} \geq \frac{\lambda^-}{N}$.

By (6) and (5) we have that $\lambda^+ \geq \beta\mu$ and $\lambda^- \leq \mu + \sqrt{2\alpha\theta}$, combining these inequalities with the definition of $\tilde{\beta}$ we have the lower bound. □

Using the upper bounds in (5) and (7) we have:

$$\lambda^+ = \tilde{\beta}\lambda^- \leq (\mu + \sqrt{2\theta\alpha})\frac{N + \beta - 1}{N}. \tag{8}$$

As the upper bound on $\tilde{\beta}$ corresponds to the case where only one connection is cut, if the right hand side of (8) is less than μ then a single cut of the connection with the biggest rate will be enough. The following lemma follows immediately by setting the right hand side of (8) less than or equal to μ and solving for θ.

Lemma 2. *If we use MarkMax-T and if*

$$\theta \leq \frac{1}{2\alpha}\frac{\mu^2(1 - \beta)^2}{(N - 1 + \beta)^2}, \tag{9}$$

then $\lambda^+ = \beta\lambda_j + \sum_{i \neq j} \lambda_i \leq \mu$ (that is after a single cut $\lambda < \mu$), where $\lambda_j = \max_i\{\lambda_i\}$.

Using the lower bound in (8) we can find a lower bound on θ so that there will be no underflow. That is the backlog is always positive and the link is fully utilized.

Lemma 3. *If we use MarkMax-T and if*

$$\theta > \frac{\mu^2(1-\zeta)^2}{2\alpha}, \tag{10}$$

where $\zeta = \frac{\beta}{1+\sqrt{2\alpha\theta}/\mu}$, *then the backlog is positive.*

Proof. We have that:

$$\begin{aligned}\lambda^+ &= \tilde{\beta}\lambda^- \geq \zeta\mu \\ &> \frac{\mu - \sqrt{2\alpha\theta}}{\mu}\mu = \mu - \sqrt{2\alpha\theta},\end{aligned}$$

where the first inequality follows from (7) and (5) and the second from (10). It is easy to see that if $\lambda^+ > \lambda^+_* = \mu - \sqrt{2\alpha\theta}$ then the backlog is always positive (see Figure (1): we want the vertex of the parabola (3) to be on the $x = 0$ axis), which completes the proof. □

We conclude with a bound that can be used as a guideline to set θ_{h}.

Lemma 4. *At time* $t = t_\theta + RTT_j$

$$x(t) \leq \theta + \sqrt{2\alpha\theta}RTT_j + \frac{\alpha}{2}RTT_j^2 \tag{11}$$

where $RTT_j = \max_i RTT_i$.

Proof. Consider a cycle that start at time t_θ then at time $t = t_\theta + RTT_i$

$$\begin{aligned}x(t) &= \theta + (\lambda^- - \mu)(t - t_\theta) + \frac{\alpha}{2}(t-t_\theta)^2 \\ &= \theta + (\lambda^- - \mu)RTT_i + \frac{\alpha}{2}RTT_i^2 \\ &\leq \theta + \sqrt{2\alpha\theta}\max_i\{RTT_i\} + \frac{\alpha}{2}\max_i\{RTT_i^2\},\end{aligned}$$

where the first equality follows from (1), and the inequality from the upper bound in (5). □

Using (11) it is possible to know by how much the queue could grow between the time the threshold θ is reached and the time the "slowest" of the connections (i.e. the one with the biggest RTT) reacts to a congestion signal.

4 Simulation Results

We have modified the NS-2 simulator in order to simulate the behavior of the proposed algorithm. We have implemented both the MarkMax-B and the MarkMax-T, referred to as MM-B and MM-T, respectively, in this section. For MM-T we only consider the last 10% of the queue (recall that for this version we are considering only the final part of the queue when determining the connection with the biggest backlog). We have compared MarkMax with the standard Drop Tail (DT) policy, by setting the queue size for DT equal to θ. For the MM case the buffer size was large enough to be considered unlimited so that we could verify that MM can stabilize the queue size.

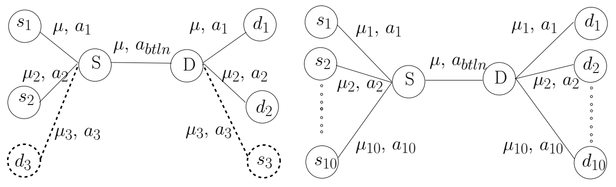
We consider three scenarios, the corresponding topologies are presented in Figure 2. Each node s_i has a TCP connection with node d_i. All the connections have a Maximum Segment Size (MSS) of 540 B. The bottleneck link is the link between the nodes S and D and has capacity μ and propagation delay a_{btlnk}. The links (s_i, S) and (D, d_i) have capacity μ_i and propagation delay a_i. For the first scenario (see Figure 2(a)) there are only two sources and two destinations while for the second scenario there is an additional TCP connection sending traffic in the opposite direction on the bottleneck link in order to introduce some variability in the flow of the acknowledgments for connections 1 and 2. The links used by this additional connection are represented as dotted lines in Figure 2(a). In the third scenario we consider 10 connections (see Figure 2(b)) with all the traffic going in one direction. In all cases only the link (S, D) uses MM while all the other links use DT.

Let $\bar{q}$ be the average queue size at the bottleneck link and $\bar{q}_i$ $(i = 1, \ldots, N)$ be the average queue sizes for the i-th connection. Using Little's formula we have that the average queueing delay at the bottleneck link is $\bar{T} = \bar{q}/\mu$. We can express the round trip time of the i-th connection as: $RTT_i = 4a_i + 2a_{\mathrm{btlnk}} + \bar{T}$, assuming the service time of each packet is negligible. Let $\delta_i \triangleq RTT_i - \bar{T} = 4a_i + 2a_{\mathrm{btlnk}}$. By increasing δ_i for some connections we model different *propagation* and *queueing* delays of multiples links that, for the sake of simplicity, are not explicitly considered.

Let t_{f} be the total simulation time. Given that all the sources start sending data at time 0 we have that the bottleneck link could transmit at most μt_{f} units of data. Let $D(t_{\mathrm{f}})$ be the total amount of data actually transmitted during the simulation so that the utilization of the link is $\rho \triangleq D(t_{\mathrm{f}})/(\mu t_{\mathrm{f}})$. Let $D_i(t_{\mathrm{f}})$ be the total amount of data received by the i-th connection so that $g_i = D_i(t_{\mathrm{f}})/t_{\mathrm{f}}$ is the corresponding goodput. To compare the fairness of different solutions we use Jain's fairness index which is defined as:

$$J = \frac{\left(\sum_{i=1}^N g_i\right)^2}{N\sum_{i=1}^N g_i^2}.$$

Note that $\frac{1}{N} \leq J \leq 1$ and that bigger values indicates greater fairness.



(a) Scenarios 1 and 2 (b) Scenario 3

Figure 2: Network topologies

4.1 Fluid Model

Using the fluid model simulator we investigate the behavior of MarkMax-B for different values of θ. In this case $\mu = 70$ Mbit/s, RTT$_1 = 12$ ms, $\alpha_i = \frac{540 \cdot 10^{-6}}{RTT_i^2}$MB/s, $i = 1, 2$. Table 1 shows the values of Jain's index and bottleneck link utilization for this case. As θ increases the utilization increases as well, due to the increase in the average backlog size. When θ is not sufficiently large the utilization is less than one due to periodic underflows. For each value of θ Jain's index decreases but it is not too far from 1.

$\frac{RTT_2}{RTT_1}$	$\theta = 60$MSS		$\theta = 240$ MSS		$\theta = 960$ MSS	
	J	ρ	J	ρ	J	ρ
3	0.9893	0.890	0.9906	0.9500	0.9815	0.9964
7	0.9874	0.892	0.9874	0.9401	0.9788	0.9990
10	0.9861	0.890	0.9869	0.9400	0.9760	0.9990
20	0.9846	0.889	0.9863	0.9440	0.9754	0.9990
50	0.9836	0.899	0.9821	0.9433	0.9664	0.9925

Table 1: Fluid Model: Jain's index, utilization.

4.2 Scenario 1

For Scenario 1 we set $\mu = 70$ Mbit/s, $\mu_1 = \mu_2 = 300$ Mbit/s, $\delta_1 = 12$ ms, $\theta = 240$ MSS, $\theta_{\mathrm{l}} = 200$ MSS, $\theta_{\mathrm{h}} = 280$ MSS, $\theta_{\mathrm{DT}} = 240$ MSS. Table 2 gives the values of Jain's index and link utilization for different values of δ_2/δ_1 and different queue management algorithms. Both MM variants outperform DT except in the first case when $\delta_2/\delta_1 = 3$. In this case Jain's index for DT is bigger but the utilization is somewhat lower. At the same time the difference between Jain's index for DT and MM is significantly large for larger values of δ_2/δ_1. Table 3 shows that the average queue size for the MM algorithms is somewhat larger than for DT. This is due to the increased link utilization obtained by MM.

We have verified that in this case the hypothesis of Lemma 2 are satisfied and in the simulations it is indeed the case that one cut is always enough to reduce the total sending rate to a value less than μ.

As the difference between MM-B and MM-T is not significant we only use MM-B in the remaining scenarios.

$\frac{\delta_2}{\delta_1}$	DT		MM-B		MM-T	
	J	ρ	J	ρ	J	ρ
3	0.9893	0.9751	0.9853	0.9999	0.9633	0.9999
7	0.7540	0.9720	0.9625	0.9999	0.9515	0.9999
10	0.5361	0.9563	0.9494	0.9999	0.9501	0.9997
20	0.5484	0.9993	0.9561	0.9994	0.9258	0.9997

Table 2: Scenario 1: Jain's index, utilization.

$\frac{\delta_2}{\delta_1}$	DT		MM-B		MM-T	
	$\bar{q}$/ B	$\bar{T}$/ms	$\bar{q}$/ B	$\bar{T}$/ms	$\bar{q}$/ B	$\bar{T}$/ms
3	78373	8.9	87257	9.9	86753	9.9
7	74802	8.5	81723	9.3	81547	9.3
10	69219	7.9	80019	9.1	79502	9.1
20	68268	7.8	74297	8.4	74189	8.4

Table 3: Scenario 1: average queue size and delay.

4.3 Scenario 2

The only difference between the first and second scenario is that there is one additional TCP connection (s_3, d_3) sending data in the opposite direction on the bottleneck link. All the parameters are the same as in scenario 1 with the only difference being that the buffer size for the DT queue between D and S (that is the queue used by the data traffic of connection 3 and the acknowledgments of connections 1 and 2) is set to 240 MSS and the $\delta_3 = \delta_2$. Table 4 shows that as in the previous scenario MM-B outperforms DT. Not surprisingly the presence of traffic competing with the acknowledgments on the (D, S) link does alter the performance of MM-B, for lower values of δ_2/δ_1 there is a slight increase in Jain's index but for higher values it decreases and the utilization is always lower than in the previous case. Most likely this is due to the fact that the presence of traffic disrupting the flow of the acknowledgments increases the round trip time.

4.4 Scenario 3

In the last scenario we have 10 connections sharing the (S, D) link and no connections using the reverse link, $\mu = 70$ Mbit/s, $\mu_i = 300$ Mbit/s, $i = 1, \ldots, 10$, $\delta_1 = 12$ ms, $\delta_{i+1} = \sqrt{2}\delta_i$, $i = 1, \ldots, 9$, $\theta = 240$ MSS, $\theta_{\mathrm{l}} = 200$ MSS, $\theta_{\mathrm{h}} = 280$ MSS, $\theta_{\mathrm{DT}} = 240$ MSS. Table 5 shows that MM-B has a significantly higher Jain's index, and slightly higher utilization, at the expenses of a moderate increase in the average queue size.

$\frac{\delta_2}{\delta_1}$	DT			MM-B		
	J	ρ	$\bar{q}$/ B	J	ρ	$\bar{q}$/ B
7	0.8561	0.9338	34443	0.9637	0.9600	41966
10	0.7769	0.9497	32174	0.9632	0.9510	39486
20	0.6910	0.9146	28699	0.9228	0.9702	41350
50	0.5244	0.9262	29021	0.8572	0.9937	50408

Table 4: Scenario 2: Jain's index, utilization and average queue size.

	J	ρ	$\bar{q}$/ B	$\bar{T}$/ms
DT	0.5848	98,91	65207	7
MM-B	0.9313	99,99	98913	11

Table 5: Scenario 3: Jain's index, utilization and average queue size and delay

5 Conclusion and Future Work

We have introduced MarkMax: a simple flow-aware AQM algorithm. We have used a fluid model to set the parameters of the algorithm as well as to analize its behaviour. We have also shown how to compute the per-flow backlog using such a model. We have simulated the two proposed variants (MarkMax-B and MarkMax-T) using NS-2, showing how they improve the fairness and link utilization compared to the standard Drop Tail algorithm.

These results are definitely promising and warrant further analysis. Of all the issues that we plan on addressing we would like to mention performance and queue stability with large number of connections and comparison between MarkMax-B and MarkMax-T. So far we have conducted simulations with up to 10 connections but it is not immediately clear if the algorithm would perform equally well with more connections. It is conceivable that, at least in some cases, cutting a single connection could no be enough to bring the total sending rate to a value smaller than μ. We would also like to determine whether MarkMax-B always outperforms MarkMax-T as indicated by the simulations we run so far or if it the situation can be reversed by properly selecting the fraction of the queue that is considered while computing the per-connection backlog in MarkMax-T.

References

[1] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of tcp/ip with stationary random loses. In *ACM SIGCOMM 2000, Stockholm, Sweden*, pages 231–242, V.30, no.4, 2000.

[2] E. Altman, R. E. Azouzi, D. Ros, and B. Tuffin. Loss strategies for competing aimd flows. *Comput. Networks*, 50(11):1799–1815, 2006.

[3] E. Altman, C. Barakat, E. Laborde, P. Brown, and D. Collange. Fairness analysis of tcp/ip. *Decision and Control, 2000. Proceedings of the 39th IEEE Conference*, 1:61–66 vol.1, 2000.

[4] E. Altman, J. Bolot, P. Nain, D. Elouadghiri, M. Erramdani, P. Brown, and D. Collange. Performance modeling of tcp/ip in a wide-area network. In *34th IEEE Conference on Decision and Control*, Dec. 12–15, 1995.

[5] E. Altman, T. Jiménez, and R. Núñez-Queija. Analysis of two competing tcp/ip connections. *Perform. Eval.*, 49(1-4):43–55, 2002.

[6] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation between short and long tcp flows: Predictability of the response time. In *IEEE INFOCOM 2004*, pages 762–773, vol 2, 2004.

[7] K. Avrachenkov, U. Ayesta, and A. Piunovskiy. Optimal choice of the buffer size in the Internet routers. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 1143–1148, Dec. 12–15, 2005.

[8] K. Avrachenkov, L. Finlay, and V. Gaitsgory. Analysis of tcp-aqm interaction via periodic optimization and linear programming: The case of sigmoidal utility function. In *NEW2AN, Also LNCS v.4003*, pages 517–529, Dec. 12–15, 2006.

[9] P. Brown. Resource sharing of tcp connections with different round trip times. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1734–1741 vol.3, 26-30 Mar 2000.

[10] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. *SIGCOMM Comput. Commun. Rev.*, 21(5):30–47, 1991.

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

[12] T. V. Lakshman and U. Madhow. The performance of tcp/ip for networks with high bandwidth-delay products and random loss. *IEEE/ACM Trans. Netw.*, 5(3):336–350, 1997.

[13] A. Mankin. Random drop congestion control. In *SIGCOMM '90: Proceedings of the ACM symposium on Communications architectures & protocols*, pages 1–7, New York, NY, USA, 1990. ACM.

[14] R. Pan, B. Prabhakar, and K. Psounis. Choke: A stateless active queue management scheme for approximating fair bandwidth allocation. In *IEEE INFOCOM 2000, Tel Aviv, Israel*, pages 942–951, Dec. 12–15, 2000.

[15] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001.

[16] S. Schenker, L. Zhang, and D. D. Clark. Some observations on the dynamics of a congestion control algorithm. *SIGCOMM Comput. Commun. Rev.*, 20(5):30–39, 1990.

[17] R. Stanojevic. *Router-based algorithms for improving Internet Quality of Service*. Phd thesis, Hamilton Institute, National University of Ireland Maynooth, 2007.

[18] R. Stanojevic and R. Shorten. Beyond choke: Stateless fair queueing. In *NETCOOP 2007, LNCS v.4465*, pages 43–53, 2007.