Low-State Fairness: Lower Bounds and Practical Enforcement

Abhimanyu Das
Dept. of CS
USC, Los Angeles
abhimand@usc.edu

Debojyoti Dutta
Dept. of Biology
USC, Los Angeles
ddutta@netweb.usc.edu

Ahmed Helmy
Dept. of EE
USC, Los Angeles
helmy@ceng.usc.edu

Ashish Goel
Dept. of MS&E
Stanford, Palo Alto
ashishg@stanford.edu

John Heidemann
USC/ISI
Marina Del Rey
johnh@isi.edu

***Abstract*— Providing approximate max-min fair bandwidth allocation among flows within a network or at a single router has been an important research problem. In this paper, we study the space complexity of fairness algorithms, and the communication complexity of distributed global fairness algorithms. We show that in order to enforce max-min fairness with bounded errors, a router must maintain per-flow state. Then we present a practical edge-marking based architecture to demonstrate the enforcement of approximate global max-min fairness for representative scenarios with multiple bottlenecks and non-responsive traffic. We validate our architecture using packet level simulations.**

I. Introduction

Fair bandwidth allocation among the different competing flows in the best effort Internet has been a traditional, well studied problem in the networking research community. As of today, TCP, which is a social, responsive, Additive-Increase Multiplicative-Decrease (AIMD) congestion control protocol, has been the dominant transport layer protocol in the Internet. However, recent studies have shown that non-congestion-reactive traffic is on the rise [1]. Resource misappropriation by high-bandwidth, non congestion-responsive or unfair flows can lead to degraded performance of congestion-reactive flows, and add to network congestion [1]. This threat is more pronounced in the face of growing UDP voice and video traffic which can quickly squeeze out other TCP([2], [3]) flows, and can even lead to a congestion collapse[4] in the Internet. Furthermore, with the ever increasing demand for bandwidth, we may not enjoy an over-provisioned best-effort Internet for too long. Hence, mechanisms to allocate bandwidth fairly among users and punishing of non-responsive flows, who deviate substantially from their fair bandwidth share, is crucial for better performance and robustness in the Internet.

Fair bandwidth allocation among flows can be imposed either locally or globally. Local fairness schemes aim to provide fairness among flows at a single router, while global fairness deals with providing network-wide fairness among all the flows in the network. Such fairness schemes can be implemented at routers in the form of either Active Queue Management (AQM) algorithms or scheduling algorithms. There are various notions of fairness [5]. In this paper we restrict ourselves to the well accepted notion of max-min fairness. Intuitively, a bandwidth allocation is max-min fair if it is not possible to give any connection or flow more bandwidth without denying an already poorer flow. Thus, max-min fairness maximizes the poorest recipient.

It is important to study the complexity of max-min fair bandwidth allocation (for both local and global fairness) algorithms. Such complexity has three components: time, space and communication. Time complexity has been studied in great detail for a long time (some recent interesting papers are [6], [7]). In this paper, we are primarily concerned with space and communication complexity. Note that communication complexity is relevant to global fairness only. It has been conjectured that it might be possible to provide approximate max-min fairness with low space complexity. We show that in general it is not possible to provide approximate max-min fairness, even locally, with low space complexity, and bounded errors. Furthermore, we conjecture that for global fairness, both the space and communication complexity may be high. However, we show that, in practice, it is possible to design schemes that approximate global fairness for certain scenarios which we consider.

As mentioned before, there is a need to provide fairness at routers. Locally fair schemes, while important, are insufficient to ensure high network wide utilization and simultaneously provide fairness among competing flows in the network. For example, unresponsive flows receiving their locally fair share of bandwidth at a link, will waste bandwidth if they are later bottlenecked at downstream nodes. Thus, it is also important to find solutions that impose network-wide fair bandwidth allocations. Note that if all end-points followed a social, responsive congestion control, and routers enforced local fairness, it would lead to high utilization as well as a globally-fair bandwidth allocation. Unfortunately, it is unreasonable to assume that all traffic will be congestion reactive. Therefore locally fair schemes such as fair queueing [8] by themselves cannot provide global or network-wide fairness.

Providing global max-min fairness has been a well explored topic. The centralized algorithm is well-known [9]. Ideally we desire a global fairness scheme that is lightweight, requires routers (both at the core as well as at the edge of the network) to maintain low state and communication complexity, and impose fair bandwidth allocations with bounded errors and high accuracy. As we will see later, such low complexity schemes are not theoretically possible, in general, due to the lower bounds that presented later in this paper. However,

we also show that lightweight schemes might be devised that approximate globally fair bandwidth allocation in certain scenarios if strict fairness bounds are not desired. For example, it may be possible to build low-state fairness schemes, e.g. when most flows are TCP friendly and there are very few misbehaving flows.

A. Our Contributions

Our contribution can be divided into the following two categories:

- **Lower bounds:** In this paper, we present the first work on lower bounds for the state needed by routers to provide approximate-min fairness. We show that in order to provide approximate max-min fairness with bounded errors among n flows, a router needs to maintain $\Omega(n)$ state. Then we conjecture that the communication complexity required to provide global fairness is equivalent to routers maintaining network-wide state. These negative results inhibit, in theory, the design of simple low-state schemes for fairness in the general case. However, as mentioned earlier, these results do not necessarily inhibit the implementation of specific algorithms for a restricted class of scenarios.
- **Practical implementation:** We study how global fairness can be imposed in practice, for certain representative scenarios using a lightweight marking based architecture. A brief overview of this architecture has been presented, in a short abstract, in [10]. In this paper we evaluate our architecture in terms of space and communication complexity.

The outline of this paper is as follows. First we present a summary of the related work in Section II. Then we present our lower bound results in Section III. This is followed by our LWD-fair marking architecture in Section IV. Finally, we conclude in Section V.

II. Related Work

Our work leverages techniques from different related research areas such as *queueing, fairness, scheduling, hashing, statistical sampling* and *traffic marking*. In this section, we will try to highlight previous work that is directly relevant to our problem of controlling misbehaving flows.

AQM: Fair Queueing (FQ) [8] at a router provides local fairness among all the flows that enter that router. However, this requires us to maintain per-flow state. A good scheme to approximate FQ have been the Core Stateless Fair Queueing architecture (CSFQ) [11], which uses flow rates embedded in packet headers. CHOKe [12] is another stateless AQM, that uses random sampling of packet queues. FRED [13] is an enhancement to RED to ensure fair buffer sharing by maintaining state for active flows at the router. Ott et al. [14] proposed SRED, which identifies candidates for high bandwidth flows from a cache of recently seen flows. RED-PD [15] uses a list of previous RED packet drops to identify misbehaving flows and control them using different drop probabilities. As mentioned earlier, none of the above can provide global fairness.

Packet Marking: Traffic marking has been an essential component of the Diffserv [16], [17] architecture too. Markers can be classified into two groups: per-flow markers and aggregate markers. Many sophisticated markers [18], [19] are per-flow based and thus not scalable, or only address TCP flows. Most current aggregate traffic markers [20], [21], [22] do not provide fairness within an aggregate. On the other hand, our CAM marker is stateless and provides approximate max-min fairness in terms of IN packets. A more complete review of the marking techniques can be found in [23].

Fairness and Admission Control: The problem of providing fairness has been a well-researched topic. For a quick introduction to the generalized issues in fairness, see [5]. Kelley [24] has been a pioneer in proportional fairness. His optimization framework involves end agents bidding with prices and the network charging the users according to all the bids. In this framework, the user optimum coincides with the system and the network optimum. Several other groups have built upon this framework. Our work is different from this entire genre of research because unlike them, we do not assume cooperative end point protocols. Also, the above scheme is stateful, unlike ours.

Charny et al. [25] have presented a distributed algorithm for providing max-min fair allocations in packet networks. However, it requires per-flow state at all the routers. Recently, Bhatnagar et al. [26] show how to compute approximate max-min fair allocation among flow groups. Their scheme requires a connection oriented model with predetermined path information for flows, in addition to per-flow state at edge nodes. Our scheme is different from the above works in several ways. First, we do not require per-flow state at the edges, or a connection-oriented paradigm. Also, since our scheme is marking based as opposed to being admission control based, we can ensure near 100% link utilizations.

Low State: The tools behind our low state claims are based upon the recent work done in streaming databases [27] and counting algorithms [28]. Our architecture needs us to quickly compute the approximate number of flows and perform flow-set intersections efficiently. at a given router. [28] propose a low-state family of bitmap algorithms for this purpose (which is what we use for our schemes). Recently, there has been a keen interest in the database community in the problems that can be clustered under the following banner: *Data processing over streams*. A good survey can be found in a recent paper [27].

Counting approximate number of flow is the same as calculating the approximate frequency moment F_0 of a data stream. In [29], Alon et al. showed that there exists a lower bound of $\Omega(n)$ space, where n is the number of flows at the router. In [30], the authors present three approximate algorithms to solve the same problem. For example, in their simplest algorithm, they show that if ϵ is the allowed error, F_0 can be calculated in $O(\frac{1}{\epsilon^2} \times \log n)$ space and time dominated by $O(\log n)$. This means that one can approximately calculate the approximate number of flows with $O(\log n)$ space for a chosen accuracy parameter.

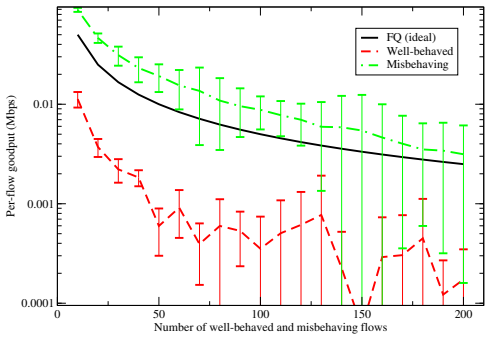

Fig. 1. CHOKe fails with large number of misbehaving flows

III. Lower bounds

In this section, we prove fundamental lower bounds for the space complexity required by a router for providing local fairness. We show that a router needs per-flow state to enforce approximate local fairness with low bounded error. The outline of the section is as follows. First we present some preliminaries in Section III-B. In Section III-C, we show that for any AQM that has a constant, but possibly different, drop rate for each flow, every randomized algorithm must maintain $\Omega(n)$ state in order to provide approximate max-min fairness. Then we show the same result for any sliding window based algorithm in Section III-D.

A. Intuition

First, let us demonstrate the intuition behind the claims made in this section by considering the performance of sample fair AQM mechanism in certain scenarios. Consider a simple bottleneck link of capacity 1Mbps, and propagation delay of 10ms. Assume that there are n well-behaved (good) and n misbehaving (bad) flows passing through this link. Ideally, the fair share should be $\frac{1}{2n}$Mbps. The good flows send at a rate equal to the fair share while the bad flows send at a much higher rate. Consider an implementation of CHOKe in NS2. We simulate the above scenario with a buffer size of 50 packets of 1000 bytes each, which is more than the bandwidth delay product. The results are shown in Figure 1. We vary n from 10 to 200. Thus, the total number of flows varies from 20 to 400. In this figure, we can clearly see that as we increase n, the accuracy of CHOKe decreases drastically. The standard deviation of bandwidth of the bad flows become larger than the bandwidth itself. The bandwidth of the good flows become very small. For example, for $n = 100$, ideal fair share gives us 5Kbps. However, the good flows get 0.3Kbps and the bad flows get 8.7Kbps. Thus, CHOKe cannot impose approximate max-min fairness for the above scenario.

We believe that the trends shown in the above experiment are not unique to CHOKe. In fact we prove that it is not possible for any *low-state AQM scheme* to impose approximate fairness with low error.

B. Preliminaries

In this section, we define our notation used in our analysis. Assume a link l with capacity C which has n connections or flows going through it. Let each connection offer λ_i amount of traffic through the link, and let μ_i be its goodput. A bandwidth allocation is max-min fair if it is feasible (i.e.$\sum_{i=1}^{n} \mu_i \leq C$), and μ_i cannot be increased while still maintaining feasibility, without decreasing μ_j for some flow j for which $j < i$. Let $R = \sum_{i=1}^{n} \lambda_i$. If $R > C$, then we can relate the *fair rate*, f, of the link l as the unique solution to the following equation:

$$\sum_{i=1}^{n} \min(f, \lambda_i) = C$$

In the max-min fair solution, each flow i gets a goodput of $\mu_i = \min(f, \lambda_i)$.

We now define a term called ϵ-fairness. An algorithm for bandwidth allocation among n flows is ϵ-fair if the bandwidth allocated to flow i is related to its offered load λ_i, the error fraction ϵ, and its max-min fair rate f by: $(1-\epsilon)\min(\lambda_i, f) \leq \mu_i \leq (1+\epsilon)\min(\lambda_i, f)$. Unless stated otherwise, we use ϵ-fairness and approximate max-min fairness interchangeably.

Information Theory: We now review some basic background in Information Theory. For a good introduction, the reader is referred to [31]. The entropy of a random variable X is defined by

$$H(X) = -\sum_{x} p(x) \log p(x)$$

Intuitively, for our purposes, it is the minimum amount of information required to represent the random variable X. Shannon's channel coding theorem [31] states that the minimum space that one needs to encode is bounded by $H(X)$ in the absence of error. For example consider an n-bit string with equal number of 0s and 1s chosen uniformly at random. Define a random variable X to denote the configuration of the above n-bit string chosen uniformly at random. Then the entropy $H(X)$ is given by $\log C^n_{n/2} = \Omega(n)$. Thus, the minimum amount of space required to represent this string is $\Omega(n)$. We will use this fact to prove our lower bounds in Section III-D.

C. Constant Drop Rates

In this section, we assume that an AQM mechanism converges to drop rates that are constant for each flow, assuming constant or Poisson traffic arrivals. We show that such a mechanism must maintain $\Omega(n)$ state to enforce bounded fairness guarantees.

Theorem 1: Any algorithm that imposes max-min fairness among n flows by converging to constant drop probabilities for each flow within any constant relative error ϵ takes $\Omega(n)$ space.

Proof: Assume there are $2n$ flows going through a router, and we need $g(n)$ space at the router, where $g(n)$ is not $\Omega(n)$. Then the total number of distinct functions that can

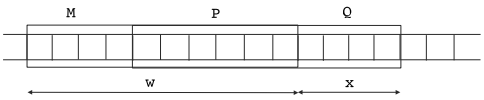

Fig. 2. Sliding window proof

be described using $g(n)$ space is $2^{g(n)}$. Now, assume that we have a scenario where there are n good flows and n bad flows. The good flows send at the fair rate, f, and the bad flows send traffic at a much higher rate of kf, where $k > 1$. Hence the number of configurations that the router needs to handle in order to impose max-min fairness using these flows is given by C_n^{2n}. Now, in order for the flows to get max-min fair share, the router needs to map the configuration to drop probabilities so that the bad flows get restricted to the fair share. Hence the router needs to use some drop function in order to do this. As we saw before, the number of drop functions can only be $2^{g(n)}$ while the total number of configurations is at least C_n^{2n}. If $C_n^{2n} > 2^{g(n)}$, then there exists at least two configurations that get mapped to the same drop function (by the familiar Pigeon Hole Principle). In that case, the router will not be able to differentiate between those two configurations. Thus, there will be at least one flow which will not get its correct fair share. Hence, $2^{g(n)} \geq C_n^{2n}$. This implies $g(n) = \log(C_n^{2n}) = \Omega(n)$. ■

D. Sliding Window

In this section, we extend the lower bound presented in the previous subsection to AQMs that are window based. That is, we show that any algorithm that imposes approximate max-min fairness with guaranteed error bounds within a window must maintain $\Omega(n)$ state.

We have the following theorem:

Theorem 2: Any algorithm that imposes ϵ-fairness among packets of n flows in a given window size W (i.e. provides fairness within every set of W consecutive packets) to within any constant relative error $\epsilon < \frac{1}{8}$ takes $\Omega(n)$ space.

Proof: Consider a sliding window of size W, and an algorithm, A, that provides ϵ-fairness within this sliding window. We provide a proof by contradiction. Assume A requires $g(n)$ state, where $g(n)$ is not $\Omega(n)$. Then we construct an encoder/decoder combination to transmit an n-bit string with an equal number of zeros and one's, using $g(n)$ bits. This violates information theory bounds for lossless coding.

Encoder Construction:

We encode the above mentioned n-bit string as follows. If the i^{th} bit of the n-bit string is 0, we create a flow with rate r pkts/s (we call it a good flow), and if the i^{th} bit is 1, we create a flow with rate kr pkts/s (such a flow will be called a bad flow). Our construction uses constant sized packets. We choose k to be much greater than 2. The total sending rate of all the flows is $R = \frac{nr(k+1)}{2}$ pkts/s. The window W therefore corresponds to $\Delta = \frac{W}{R}$ seconds of the aggregate flow rate R. Let $x = r\Delta$. Thus, every window of W packets contains exactly $x = r\Delta$ packets of each of the $n/2$ good flows and $kx = kr\Delta$ packets of each of the $n/2$ bad flows. Set the capacity of the black box to be $C = 2nr$ pkts/sec. Thus, among the window W of packets, only $C\Delta = 2nx$ packets are accepted, and the rest are dropped. The max-min fair rate f of the router therefore corresponds to $3x$ accepted packets per flow within the window W. Now let us insert a total of W packets from these flows into A in a weighted round robin fashion, with weights 1 and k corresponding to good and bad flows respectively. The state of the algorithm, is sent as the encoded state. Thus the size of the code sent by the encoder is $g(n)$, which is not $\Omega(n)$ by assumption.

Decoder Construction: We make n copies of the algorithm A's state (received from the encoder) for decoding the n bits. For the i^{th} copy, we decode bit i as follows. We send x new packets labeled i through A. Then we observe how many of these packets are accepted, and decide the value of the bit accordingly.

Consider the last W packets seen at the encoder (before the x packets are inserted at the decoder, as above). Let us partition these W packets into two sets, M and P, such that $|M| = x$ packets and $|P| = W - x$ packets. Also, let the set of x new packets sent at the decoder be denoted by Q. By our assumption, the algorithm A provides ϵ-fairness within every window of W consecutive packets seen. In particular, this is true for each of the two windows formed by the consecutive sets MP and PQ, where $MP = M \cup P$ and $PQ = P \cup Q$. This is shown in Figure 2. Consider the window MP, for example. Note that the fair share of a good flow within MP is x.

Flow i is bad: We will find out the maximum number of accepted packets of flow i in Q denoted by maxQ_i. Consider the window MP. Now, the ideal fair share of this flow in the window MP is given by

$$\text{FS}_{\text{MP}}^{\text{bad}} = \frac{2nx - \frac{n}{2}x}{\frac{n}{2}} = 3x.$$

Also, the maximum possible number of packets of i accepted in M can be obtained by assuming that all packets of flow i in M are accepted; this is given by

$$\text{maxM}_i^{\text{bad}} = \left(\frac{k}{k+1}\right)\left(\frac{x}{\frac{n}{2}}\right) = \frac{2kx}{(k+1)n}.$$

Therefore the minimum possible number of accepted packets of i in P, $\text{minP}_i^{\text{bad}}$ is given by

$$\text{minP}_i^{\text{bad}} = \text{FS}_{\text{MP}}^{\text{bad}}(1-\epsilon) - \text{maxM}_i^{\text{bad}} = 3x(1-\epsilon) - \frac{2kx}{(k+1)n}.$$

Now consider the window PQ. The ideal fair share of this flow in the window PQ is given by

$$\text{FS}_{\text{PQ}}^{\text{bad}} = \frac{2nx - \frac{W-x}{k+1}}{\frac{n}{2}} = 3x + \frac{2x}{(k+1)n}.$$

Therefore, maxQ_i is given by the following equation:

$$\text{maxQ}_i^{\text{bad}} = \text{FS}_{\text{PQ}}^{\text{bad}}(1+\epsilon) - \text{minP}_i^{\text{bad}} = 6x\epsilon + \frac{2x\epsilon}{(k+1)n} + \frac{2x}{n}.$$

Flow i is good: In this case, we will find out the minimum possible number of accepted packets of flow i in Q denoted by $\text{minQ}_i^{\text{good}}$. Consider the window PQ The maximum possible number of accepted packets of i in P can be obtained by assuming that all packets of flow i in P are accepted; this is given by

$$\text{maxP}_i^{\text{good}} = \frac{W-x}{(k+1)\frac{n}{2}}.$$

The ideal fair share of this flow in the window PQ is given by

$$\text{FS}_{\text{PQ}}^{\text{good}} = x + \frac{W-x}{(k+1)\frac{n}{2}}.$$

Therefore, $\text{minQ}_i^{\text{good}}$ is given by the following equation:

$$\begin{aligned}\text{minQ}_i^{\text{good}} &= \text{FS}_{\text{PQ}}^{\text{good}}(1-\epsilon) - \text{maxP}_i^{\text{good}} \\ &= x(1-\epsilon) - \frac{(W-x)\epsilon}{(k+1)\frac{n}{2}} \\ &= x - 2x\epsilon + \frac{2x\epsilon}{(k+1)n}.\end{aligned}$$

Now, if $\text{minQ}_i^{\text{good}} > \text{maxQ}_i^{\text{bad}}$, then we can clearly decode flow i as good or bad, based on the number packets that are accepted out of the x packets in Q. This simplifies to the following condition

$$x - 2x\epsilon + \frac{2x\epsilon}{n(k+1)} > 6x\epsilon + \frac{2x}{n} + \frac{2x\epsilon}{(k+1)n},$$

or $1 - 2/n > 8\epsilon$. Thus, for large n, if $\epsilon < 1/8$, we can decode all the n bits without error. This violates the lower bounds in coding in the following way. The size of the minimum code is given by the entropy of the system. Now the entropy of the n-bit string is given by $\log C_{n/2}^n = \Omega(n)$. But by assumption, the code size is $g(n)$, which is not $\Omega(n)$. This is a contradiction. ■

Note that we used very simple estimates for parameters such as x and k. By manipulating the parameters, it is possible to improve the bounds on ϵ.

From the above theorem, it is easy to see the following corollary:

Corollary 1: Any algorithm that for globally max-min fair bandwidth allocation needs $\Omega(n_i)$ state at each individual router, where n_i is the number of flows that pass through router i in the network.

In a globally max-min fair algorithm, a router will need to communicate certain amount of information to other routers. A natural question that follows is how much routers should communicate in addition to maintaining $\Omega(n_i)$ state. We conjecture that routers need to communicate a large amount of information. In the worst case, it may be $\bigcup n_i$ amount of information. This is equivalent to routers maintaining global state. The intuition comes from a recent result in data streams [32] which says that set operations such as difference or union of two data streams A, B require at least $\Omega(\frac{|AUB|}{|AopB|})$, where op is either difference or intersection. Thus we have the following conjecture:

Conjecture 1: Any algorithm that for globally max-min fair bandwidth allocation needs to communicate $\Omega(\bigcup_i n_i)$ state at each individual router, where n_i is the number of flows that pass through router i in the network.

E. Discussions

In this section, we proved that in order for any algorithm to impose max-min fairness with high accuracy, it needs to maintain per-flow state. This might seem counter intuitive with the presence of several fair AQMs such as CHOKe, for example. However, we must note that all the previous work on low-state AQMs have considered a small number of flows (less than a thousand). Most AQMs have not done a study of how their accuracy drops when the state maintained by the routers are less in comparison to the number of flows. In contrast, our result does not characterize any particular AQM. The result shows a lower bound by considering an adversarial scenario where every algorithm will fail. Thus, it is fair to ask questions such as: whether there are similar lower bounds for low state fair AQMs that are only designed for practical scenarios.

Although we prove that, in general, per-flow state complexity for imposing fairness with bounded errors is necessary, it is possible to have practical implementations that provide reasonable fair bandwidth allocations for practical scenarios. We will show one such fairness architecture in the next section.

IV. Low-state Enforcement

The pessimistic lower bounds presented in the previous section are for enforcing max-min fairness with low error bounds. In practice, we can do better than the above results if we relax our strict error requirements. In this section, we will show one simple mechanism to obtain approximate global fairness without drastically changing the network infrastructure. We present our lightweight, low-state, packet-marking based (LWD-fair) architecture to impose fairness. A preliminary version of this architecture has been presented, as a short abstract, in [10]. Here we expand upon the abstract and present a detailed evaluation of the architecture.

A. Our Approach

The core components of our architecture are a set of traffic markers at the edges (based on low state or stateless fair AQMs) that mark packets in an approximate fair fashion (in practice), and a distributed marking algorithm that utilizes these AQM-based markers to classify packets into two classes, IN and OUT, using low-state feedback obtained from the congested routers (with the help of summary data structures such as bit vectors or sketches) to allocate approximately global max-min fair IN token rates to all the flows in a network. During congestion, the congested router uses RIO (RED IN/OUT) as the AQM policy and preferentially drops the OUT packets. See Fig. 3 for a summary of our architecXSture.

To design efficient traffic markers which mark flows of an aggregate fairly, we leverage the queueing and dropping policies of existing low-state approximately fair AQM algorithms, and we call these aggregate packet markers **AQM-based markers**. These markers mimic well-explored, approximately fair AQM schemes. Note that as we have just proved, it is not possible, in theory, to design fairness algorithms with low-state and bounded errors. However, the fair AQMs, that form the basis of our markers, have been shown, empirically, to perform adequately for the scenarios we consider in this paper. It was previously shown [23] that the problem of fair token distribution of IN tokens among flows at a marker among an aggregate of flows, using a token bucket traffic specification, can be viewed as being equivalent to fair buffer management and scheduling of packets at a queue of the corresponding AQM policy. For example, we adapt CHOKe [12] to design a marker called CAM (CHOKed Aggregate Marker).

When a link is close to being congested, the router (for the link) starts inserting the flow id of all the packets it sees into a summary data structure (DS). For example, we could set the bit of a bit-vector that the flow hashes into or update a *sketch* [27]. On congestion, the congested router sends the DS (e.g. bit vector) as well as its bandwidth to all the ingress points in the network. It obtains the DS from all the congested links and calculates the approximate globally fair marking rate corresponding to each congested link as descrribed in Algorithm 1. We design a general purpose architecture that works independent of the particular data structure used. We can leverage flow summary data structures that need at most $poly - \log n$ space [33], [28]. However, using such data structures may not guarantee strict bounds on the fair rate obtained. Using our LWD-fair algorithm, each packet is then marked with a target rate that is the minimum of the fair target rates for all the links that the packet traverses through (such membership information is obtained from the summary data structures we use). Each edge router has one AQM based marker [23] (CAM, which is based on CHOKe, for example) for each bottleneck. Thus, the packet marking is performed by the marker for the bottleneck corresponding to the lowest target rate seen by the flow.

B. LWD-fair marker

In this section, we describe our LWD-Fair algorithm to obtain a globally fair distribution of IN tokens among flows. Intuitively, our LWD-fair marker is similar to that presented in Bertsekas et.al. [9] for calculating globally max-min fair rates. However, using summary data structures such as sketches and bitmaps [28], we may not need to keep per-flow state information as in traditional fairness calculation algorithms. The LWD-fair algorithm iterates over all the bottleneck markers by choosing the most congested bottleneck first and fixing its fair marking rate, and then adjusting the data structures for other bottlenecks.

The basic foundation of the algorithm is as follows: At each iteration, the bottleneck-link L with the smallest local fair rate is chosen and its marking rate fixed. The remaining links' available token capacity then have to be reduced depending on how many flows they have in common with Link L.When all bottleneck-links have been covered, their marking-rates corresponding to globally max-min fair bandwidth allocation. To make our algorithm independent of the data structures used for flow information feedback, we assume that feedback from all bottleneck-ed links l is received by summary data structures $DS[l]$ containing information about the flows at those links. All we expect is that these summary data structures can support five set operations: Cardinality, Intersection, Union, Set Difference, and Set Membership.

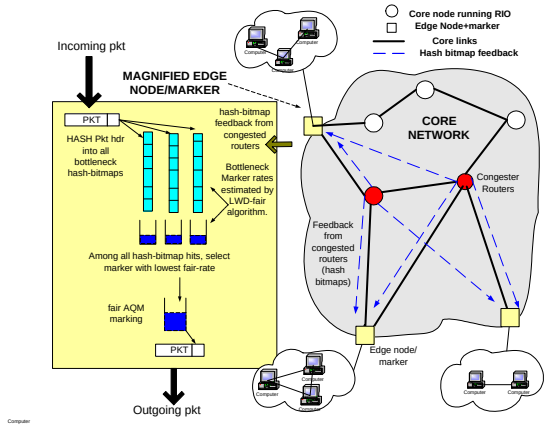

Fig. 3. LWD-fair algorithm and our marking framework

Algorithm 1: LWD-Fair Algorithm

Data : $local$: edge's incoming link;
$Markerrate[i]$: tokenbucket rate for bottleneck i's marker;
$Marker[i]$: tokenbucket marker for bottleneck i;
$DS[i]$: flow summary received from bottleneck i;
$Cardinality(DS[i])$: num of flows calculated from summary DS[i] for bottleneck i;
$Bw[i]$: link bandwidth of bottleneck i;
M: set of bottleneck links that sent feedback;
P: temporary flow datastructure initially empty;
$N[i]$: initialized to Cardinality(DS[i]);
$Intersect(DS[i], DS[j])$: datastructure containing the intersection of flows of bottlenecks i and j);
$Union(DS[i], DS[j])$: datastructure containing the union of flows belonging to bottlenecks i and j);
$Difference(DS[i], DS[j])$ datastructure containing the set difference of flows belonging to bottlenecks i and j;
$Member(f, DS[i])$: boolean variable that is true if flow f passes through bottleneck i;

if *event == new summary-datastructure feedback* **then**
  cnt=0;
  while *M is not empty* **do**
    foreach *i such that i is in M* **do**
      $est\_rate[i] = \frac{Bw[i]}{N[i]}$;
    tightlink = link s.t. $est\_rate[link]$ is min in M;
    $fairrate[tightlink]$=$est\_rate[tightlink]$;
    remove tightlink from M;
    P = Union(P, DS[tightlink]) //add flows of tightlink to P;
    foreach *j in M* **do**
      $temp$ =;
      $cardinality(Difference(Intersect(DS[tightlink], DS[j]), P))$;
      $Bw[j] = Bw[j] - temp * fairrate[tlink]$;
      $N[j] = N[j] - temp$;
    cnt++;
  $Markerrate[i] =$
  $Cardinality(Intersect(DS[local], DS[i])) * fairrate[i]$

if *event == recvd new pkt(p)* **then**
  MarkerSet = All links l in M such that Member(p,DS[l])=true;
  MarkerBottleneck = m $\in$ Markerset such that fairrate[m] is minimum among all links in MarkerSet;
  Mark(p,Marker[m]);

C. Low State

It may not scalable for a congested link to identify, store and transmit all its active flow-id's during congestion. By using summary data structures, such as multi-resolution bitmaps [28], [34], from the congested routers instead of per-flow information, our LWD-fair algorithm can reduce the state and communication required in the centralized global max-min algorithm. However, the choice of the data structure dictates the accuracy of the algorithm. We therefore maintain a summary data structure per link at each router, which requires low update time per arriving packet (for example, each packet is hashed on its flow-id to update a single bit in the bitmap [34], [28]).

For example, Multi-resolution bitmaps are an extension of direct bitmaps that rely on updating a bit in a hash table for each packet. If N is number of flows, and ϵ is allowed error, the number of bits to store flow information in this DS = $O(\frac{logN\epsilon^2}{\epsilon^2})$. Using different sampling factors for different areas of the bitmap, Varghese et.al. [28], [34] show that they can significantly improve on the accuracy of direct bitmaps without much extra memory requirements. They also show that using multi-resolution bitmaps with 8k bits, average errors for counting up to 1000000 flows are only 3%. However, these bitmaps also introduce errors especially in set operations. A direct bitmap, on the other hand, is simple, more accurate, than that of a multi-resolution bitmap; howeverm, it requires more space to implement. Thus, the accuracy depends on the amount of state required. The choice of the summary DS dictates the tradeoffs in errors versus the state and communication complexity required.

D. Signaling Overheads

In this section, we discuss the signalling overheads for the proposed feedback information generated by bottleneck routers, that is needed by our LWD-fair algorithm. By comparing the bandwidth estimates of the feedback used in our scheme versus OSPF overheads for a given network topology, we argue that the bandwidth overheads of our scheme is not significant.

We first quantify the overhead of the LWD-fair algorithm in a general setting and then we show that for practical settings, the overhead is not much more than that required by intra-domain routing protocols such as OSPF. Without lack of generality, we consider the simplest and the most inefficient summary data structure ie. the direct bit map. This gives us an idea of the worst case performance bounds.

Note that the DS feedback is generated by a router only when one of its links is congested; that is, the link utilization is above a certain congestion threshold. A simple way for a router to detect if any of its links are congested could be to look at the EWMA (exponentially weighted moving average) of the queue length for the link. For example, if the link's EWMA queue length is greater than a certain threshold (say 70% of the total queue size) the link could be considered congested for our purposes of sending feedback to the edge routers. Additionally, to reduce the frequency of feedback sent for a congested link when the number and constitution of flows at that link are constantly changing, we propose to have a dampening mechanism in place. A DS feedback will be sent only when the current DS differs from the previously sent DS by a non-trivial percentage.

In a typical scenario, we assume (for the purposes of calculating an estimate of the feedback bandwidth requirements) that once congestion has been detected for a link, and the corresponding DS feedback has been generated for that link, the constitution of flows at the link during the period of that particular congestion event will not change significantly. Hence there will be approximately one DS feedback generated by the router per congestion event at a link. To the best of our knowledge, there is no published study of the frequency of congestion events. For the purposes of signalling overhead comparison, we estimate the feedback bandwidth requirements over a range of feedback frequency values varying from 30 seconds to 3 hours.

Consider a network $G(V, E)$. Let n be the number of nodes and m be the number of links. Let ρ be the fraction of the links that are congested. Assume that the frequency of feedback information sent per link is λ. Also suppose that there are N flows and we desire an error in the summary data structures that is no more than ϵ. Consider the worst case scenario of all the congested routers sending feedback at the same time. For a direct bitmap, the space requirement, and, hence, the feedback payload is given in [34] by $\zeta_{\text{lwd}} = \frac{N}{\log N\epsilon^2}$. Thus, the total average overhead due to feedback on a single link ,η is given by

$$\eta_{\text{lwd}} = \lambda\rho n\left(\frac{N}{\log N\epsilon^2}\right) \tag{1}$$

To argue that the overheads due to LWD-fair are reasonable, let us compare them with that of OSPF. OSPF is tuned, engineered, and deployed such that its overheads are not significant. Thus, it is reasonable to use OSPF as a point of comparison.

For OSPF[35], the major overheads are due to HELLO messages and LSA(Link State Advertisement)-updates. Hello messages are typically exchanged over links, while LSA-updates are typically flooded every 30mins. Using reasonable values for OSPF parameters (HELLO messages of 50 bytes every 10s and 200 bytes per router-LSA every 30 mins), we can approximate the average OSPF overhead(due to router LSAs and HELLO messages alone) on any link in a single OSPF area of n routers, η_{ospf}, which is given by

$$\eta_{\text{ospf}} = \frac{1}{30\text{mins}}n * 200\text{bytes} + \frac{50}{10\text{sec}}\text{bytes}. \tag{2}$$

Clearly, if we fix parameters such as ρ, λ, N and ϵ, asymptotic overheads for both LWD-fair as well as OSPF are comparable.

Now consider practical values of each of the above parameters. Consider an ISP of 100 nodes where each routers handles approximately 100K flows and around 10% of all links in the network are congested. First, we assume that feedback per link needs to be generated every 30 mins. We will tolerate a maximum error of 10% for our bitmap queries. Now, from

TABLE I

FEEDBACK OVERHEADS WITH 100000 FLOWS, FOR A RANGE OF FEEDBACK FREQUENCIES.

Feedback Frequency	Bandwidth Overhead
3 hrs	1.17B/s
1 hr	3.5B/s
30 mins	7B/s
3 mins	70B/s
1 min	210B/s
30 sec	420B/s

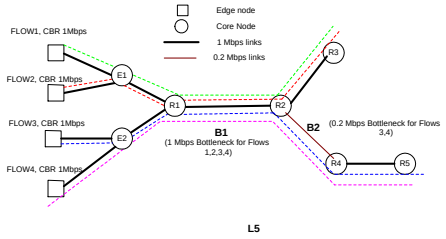

Fig. 4. The multiple bottleneck simulation topology

CBR Flow Throughputs (Mbps)	*Flow 1*	*Flow 2*	*Flow 3*	*Flow 4*
Theoretical global max-min fair rates	0.4	0.4	0.1	0.1
LWD-fair algorithm with CAM marking	0.42	0.41	0.083	0.082
Fair Queueing at each node	0.249	0.249	0.099	0.099

Fig. 5. LWD-Fair - all UDP

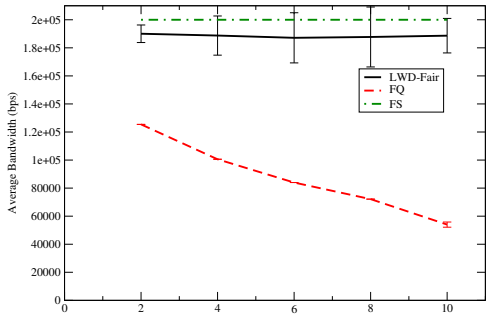

Fig. 6. LWD-Fair - TCP+UDP

Equation 1, $\eta_{\text{lwd}} = 7$ B/s and $\eta_{\text{ospf}} = 16$ B/s. Even if we assume that 50% links are congested, $\eta_{\text{lwd}} = 35$B/s which is clearly not too far off. If we now assume that the feedback frequency is around 3 minutes, η_{lwd} is still 70 B/s, (as shown in the table) which is comparable to OSPF. Note that we have used pessimistic estimates for calculating the overheads of LWD-fair algorithm (direct bitmaps). The space requirements are significantly better if we use multiresolution bitmaps, for example. Thus, we believe that for most practical scenarios our LWD-fair algorithm will not add significant additional bandwidth overhead.

E. Results

In this section, we evaluate our framework in a step-by-step fashion studying the trade offs and the relationship with the performance. We study multiple bottleneck scenarios based on two topologies, the dumbell and the parking lot. We use packet level simulations for our study and we model misbehaving flows as high-bandwidth CBR flows over UDP.

1) Experimental Setup: We conducted our simulations using the ns-2 [36] network simulator. We assumed simple direct bitmaps for the summary data structures. The main objective of the evaluation was to check the efficacy of the marking based architecture. The architecture is meant to be independent of both the feedback as well as the summary data structures to be used. In this evaluation, we consider only multiple bottlenecked scenarios and omit the results for the simpler case with single bottlnecks.

2) Simple Multiple Bottlenecks: First we study a network topology with multiple bottlenecks and two edges, as shown in Figure 4. Then we evaluate our architecture for a more complex parking lot topology.

We first consider a simple example with four UDP flows,. All the flows are 1Mbps UDP sources, and there are two bottlenecks - a 1Mbps link and a 0.2Mbps link. We compare the end to end throughput obtained by the UDP flows in our framework, with that obtained by running Fair Queueing at all nodes. We observe in Figure 5, that using our LWD-fair marking algorithm, the bandwidth obtained by the flows actually comes very close to that theoretically predicted by global max-min fairness constraints, while local fair-queueing will not provide global fairness, and will actually provide lower total throughput than our scheme. Thus we see that for non-congestion reactive flows, fair-queuing does not provide high network utilization, while LWD-fair marking does. A corollary of this result is that since our scheme here approximates global fairness, all misbehaving flows in the network are therefore automatically curbed to their fair levels.

We now consider a more complex case in the same topology as earlier, but using a mixture of TCP and UDP flows in the network. We use a fixed set of 4 TCP aggregating at edge $E1$ and flowing out through node $R3$, while we vary the number of 1Mbps UDP flows (from 2 to 10), which aggregate at edge $E2$ and leave from $R5$. There are thus two bottlenecks, the 1Mbps bottleneck for all flows, and the subsequent 0.2

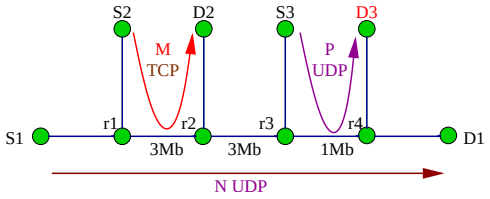

Fig. 7. The parking lot topology for studying multiple bottlenecks

Mbps bottleneck for the UDP flows later. As a measure of network utilization and fairness in this scenario, we compare the average TCP throughput obtained, in our scheme, Fair Queueing (FQ), and theoretically predicted global fairness estimates, or the global Fair Share (FS). The globally fair values for the TCP flows should be $(1-0.2)/4 = 0.2Mbps$ since the UDP flows are bottlenecked later at a 0.2Mbps link. The results obtained in our scheme are very similar to the theoretical global max-min values. However with FQ, we see that the TCP performance degrades progressively as the number of UDP flows increase, due to FQ's attempt to provide local fairness in spite of the fact that the UDP flows will not be able to use their allocated fair-share further ahead. We therefore illustrate that in the absence of global fairness, locally fair router schemes may not provide high network utilization if some flows are non congestion-reactive. The graphs in Figure 6 plots the average goodput of the TCP flows as the number of UDP flows increases. Ideally, the total UDP goodput should not exceed 0.2Mbps due to the bottleneck at the link $R2-R4$. This implies that the TCP goodput should not vary as the number of UDP flows are increased. However, Fair Queueing at $R1$ is unaware of the bottleneck at the link $R2-R4$ for the UDP flows. Hence it decreases the goodput of the TCP flows as the number of UDP flows increases. This graph indicates that our LWD-fair scheme along with CAM markers does not face this problem.

3) Parking lot topology: Now let us consider the parking lot toplogy as shown in Fig. 7. The nodes S_i are the sources while the destinations are D_i. There are three groups of flows, namely (S_1, D_1), (S_2, D_2) and (S_3, D_3) with N, M and P flows respectively. Apart from the bandwidths shown, all the other links have 10Mb/s or more.

In our experiment we keep $M = P = 5$. Thus we have 5 TCP flows from S_2 to D_2 and 5 UDP flows from S_1 to D_1. We vary N, the number of UDP flows from S_1 to D_1. We compare the average throughput of the TCP flows with the goodput obtained if FQ was used. The results are depicted in Fig. 8, where we plot the average bandwidth of the TCP flows from S_2, D_2. Note that for all the M flows, the bottleneck is link (r_2, r_3). Now, the ideal centralised fair share algorithm would give each of these N, P flows $\frac{1}{N+P}$Mbps. Thus the N TCP flows would each get

$$BW_{TCP}^{FS} = \frac{3 - \frac{N}{N+P}}{M}\text{Mbps} \quad (3)$$

On the other hand, if we use FQ at all routers, we will not achieve the globally fair share (FS) rates because at

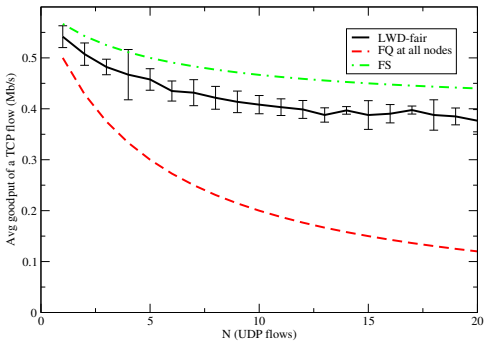

Fig. 8. Performance of TCP flows with LWD-fair marking is very close to idea fair share

router r_1, FQ would give each flow equal share and every flow would get $\frac{3}{M+N}$. Hence there would be a drop in the network utilization. Clearly, our LWD-fair does not face this problem as it is has the feedback information of the flows from the bottlenecks. When the packets to D_1 appear at S_1, it is marked with the target rate of the most congested link it encounters, which in this case is r_3r_4. The M TCP flows get marked with the target rate of the router at r_1 which is approximately equal to Equation 3. This rate is due to our LWD-fair algorithm and the error in this rate calculation depends on the summary data structure used. Thus the IN token allocation is approximately max-min fair. On congestion, RIO drops all the OUT packets and this leads to an approximate globally max-min fair bandwidth allocation.

Finally, we study the effects of the errors in the summary structures on the performance of our LWD-fair algorithm. The sources of error in this architecture primarily stems from the errors in summary data structures used. Now, this error is difficult to characterize theoretically. Low state summaries might cause a reduction in accuracy of the fair rate calculations. In our evaluation, we consider accurate bitmaps and only two congested routers. In order to determine the robustness of our LWD-fair algorithm, we artificially induce error in the fair rates calculated for each router by the markers. This gives us a metric to study how robust our algorithm is to errors.

We fix the parameters M, N, P in the above experiment to be 5 each. Then we introduce errors in the approximate summary DSs sent by the congested nodes. This is modelled as an error in the number of flows bottlenecked at that router. We vary the error from 0.02 to 0.5 and plot the average bandwidth in Fig. 9. We find that the error introduced in the end-to-end goodput of the TCP flows is only 12% (compared to LWD-performance without error) as we increased the error in the summaries to 50%. Thus, for the practical scenarios we consider, it seems that our algorithm might be robust to errors introduced by summary DSs.

Thus we have shown that for representative scenarios considered in this paper, our LWD-fair markers along with

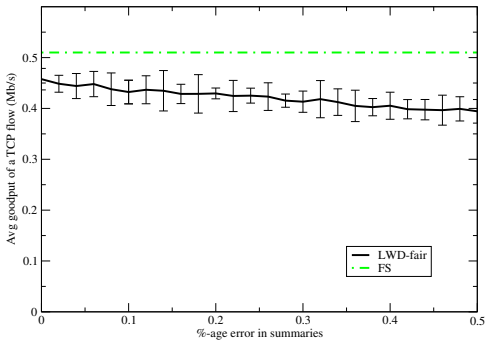

Fig. 9. Performance of TCP flows with LWD-fair marking is robust to the error in feedback

feedback and RIO dropping are effective in dampening of misbehaving flows and enforcing a network-wide approximately max-min fair bandwidth allocation.

4) Discussions: In this section, we introduced a lightweight architecture that can impose an approximate globally max-min fair bandwidth allocation and provide efficient network-wide bandwidth utilization. Our framework consists of marking packets at edges using a LWD-fair marking algorithm that uses low-state feedback from core router and AQM-based marking to allocate bandwidth to flows in an approximately globally or network wide max-min fair manner. Using detailed packet level simulation, we observe that in representative topologies with multiple bottlenecks and non-responsive traffic, our architecture can improve TCP performance in the presence of misbehaving flows, by an order of magnitude. We also show that our scheme outperforms FQ in terms of network utilization and global fairness, and approaches the theoretical global max-min fairness levels.

V. Conclusions and Future Work

In this paper, we studied the space and communication complexity of the well known problem of providing fairness (both local and global) among different flows within a network, and a practical scheme to implement global fairness for certain scenarios. We motivated the global fairness problem by observing that even if we deploy fair queueing at each router, we may not be able to provide fairness and maintain high utilization simultaneously.

The main contribution of this paper is a lower bound for the amount of space required by a router to provide approximate max-min fairness with bounded errors. We are unaware of any other work that has explored the space complexity of fairness algorithms. We show that in order to enforce max-min fairness with bounded errors, a router must maintain per-flow state. This result translates to lower bounds for global max-min fairness as well. We also conjecture that each router must communicate per-flow state for enforcing global fairness with bounded errors. Then, we present an edge-marking based architecture to demonstrate the practical enforcement of approximate global max-min fairness for representative scenarios with multiple bottlenecks and non-responsive traffic.

One potential direction for future work is to characterize scenarios for which the space complexity to impose fairness may be $o(n)$, where n is the number of flows. Such a characterization will be of great help to network designers. Another interesting theoretical direction is to prove our conjecture for the communication complexity of the global fairness problem.

We are encouraged by the initial results of the LWD-fair architecture, and we plan to determine the space versus accuracy trade-offs when different summary data structures are used. We know that though we cannot improve on the $\Omega(n)$ bound for the space complexity in the general case, we hope to obtain better bounds for specific scenarios with skewed distributions of flow rates, and use this to design better AQM mechanisms.

References

[1] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[2] J. Postel, "Transmission control protocol," Internet Request for Comments RFC 793, September 1981.

[3] V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988*, vol. 18, 4, pp. 314–329, 1988.

[4] J. Nagle, "Congestion control in ip/tcp internetworks," 1984.

[5] R. Bhargava, A. Goel, and A. Meyerson, "Using approximate majorization to characterize protocol fairness," in *SIGMETRICS/Performance*, 2001, pp. 330–331.

[6] J. Xu and R. J. Lipton, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. 2002, pp. 279–292, ACM Press.

[7] R. R. Kompella and G. Varghese, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms," in *NOSSDAV*, 2004.

[8] A. Demers, S. Keshav, and S.J. Shenker, "Analysis and simulation of a fair queueing algorithm," *Sigcomm*, 1989.

[9] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, second edition, 1991.

[10] A. Das, D. Dutta, and A. Helmy, "A low-state packet marking framework for Approximate Fair Bandwidth Allocation," *IEEE Communication Letters (to appear)*, 2004.

[11] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," *Sigcomm*, 1998.

[12] R. Pan, B. Prabhakar, and K. Psounis, "A stateless active queue management scheme for approximating fair bandwidth allocation," *IEEE INFOCOM 2000*, 2000.

[13] D. Lin and R. Morris, "Dynamics of random early detection," *SIGCOMM '97*, pp. 127–137, september 1997.

[14] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proceedings of INFOCOM*, 1999, vol. 3, pp. 1346–1355.

[15] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," 2001.

[16] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, 1998.

[17] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475*, 1998.

[18] I. Yeom and A. L. Narasimha Reddy, "Adaptive marking for aggregated flows," *Globecomm*, 2001.

[19] A. Feroz, A. Rao, and S. Kalyanaraman, "A tcp-friendly traffic marker for ip differentiated services," *Proc. of the IEEE/IFIP Eighth International Workshop on Quality of Service - IWQoS*, 2000.

[20] J. Heinanen and R. Guerin, "A single rate three color marker," *RFC 2697*, 1999.
[21] J. Heinanen and R. Guerin, "A two rate three color marker," *RFC 2698*, 1999.
[22] H. Su and Mohammed Atiquzzaman, "Itswtcm: A new aggregate marker to improve fairness in difserv," *Globecomm*, 2001.
[23] A. Das, D. Dutta, and A. Helmy, "Fair stateless aggregate marking techniques using active queue management techniques," in *IEEE/IFIP MMNS*, October 2002.
[24] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," in *Journal of the Operational Research Society*, 1998, vol. 49.
[25] A. Charny, D. Clark, and R. Jain, "Congestion control with explicit rate indication," *In Proceedings of IEEE International Conference on Communications*, pp. 1954–1963, 1995.
[26] Sudeept Bhatnagar and Badri Nath, "e-fairness: A trade-off between overhead and max-min fairness," in *IEEE ICC*, 2003.
[27] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2002, pp. 1–16, ACM Press.
[28] Cristian Estan, George Varghese, and Mike Fisk, "Bitmap algorithms for counting active flows on high speed links," 2003.
[29] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Journal of System Sciences*, 1996, pp. 20–29.
[30] Z. Bar-Yossef, T. S. Jayaram, S. Ravi Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *RANDOM 2002*, 2002.
[31] T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, second edition, 1991.
[32] S. Ganguly, M. Garofalakis, and R. Rastogi, "Processing set expressions over continuous update streams.," *ACM Sigmod*, June 2003.
[33] G. Manku and R. Motwani, "Approximate frequency counts over data streams," in *In Proceedings of the 28th International Conference on Very Large Data Bases*, July 2002.
[34] Cristian Estan, George Varghese, and Mike Fisk, "Bitmap algorithms for counting active flows on high speed links," *ACM IMC*, October 2003.
[35] J. Moy, "OSPF version 2, RFC2328," apr 1998.
[36] UCB/LBNL/VINT, "The NS2 network simulator, available at http://www.isi.edu/nsnam/ns/.," .