Approximate Fairness Through Differential Dropping (summary)

Rong Pan, Stanford University
Lee Breslau, AT&T Labs-Research
Balaji Prabhakar, Stanford University
Scott Shenker, ACIRI

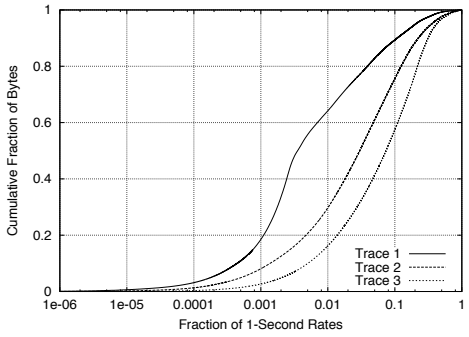

Fig. 1. Complementary Distribution of 1-Second Rates

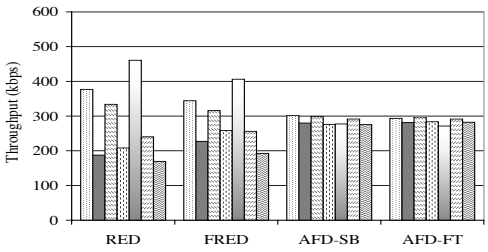

Fig. 2. Mixed Traffic - throughput

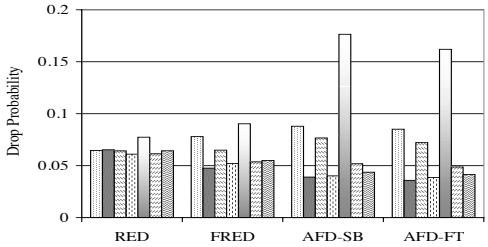

Fig. 3. Mixed Traffic - drop probability

Many researchers have argued that the Internet architecture would be more robust and more accommodating of heterogeneity if routers allocated bandwidth fairly. Most of the mechanisms proposed to accomplish this fall into two general categories, each with their own drawbacks. The first category, which includes Fair Queueing (FQ) and its many variants, involves complicated packet scheduling algorithms that may not be inexpensively implementable at extremely high speeds. The algorithms in the second category, active queue management schemes with modification to improve fairness (*e.g.*, FRED, SFB), are easy to implement and are much fairer than the original RED design, but are not intended to provide max-min fairness among a large population of flows . Our research focuses on the design of a router mechanism that achieves approximately fair bandwidth allocations with relatively low implementation complexity.[1]

There are three basic goals of our design. First, the resulting bandwidth allocations should be approximately max-min fair. We evaluate fairness over relatively long time scales, on the order of several roundtrip times. Second, the design should be easily implementable at high speeds. We limit ourselves to FIFO packet scheduling algorithms with probabilistic drop-on-arrival, and use only a small amount of state (compared to the packet buffers) to make these dropping decisions. Finally, the algorithm must have active queue management.

To achieve these goals, we propose an algorithm called Approximate Fair Dropping (AFD). It uses a FIFO queue with probabilistic drop-on-arrival like RED. However, these dropping decisions are based not only on the queue size but also on the flow's current sending rate r_i. Under congestion, a flow's packet is discarded with a probability $d_i = (1 - \frac{r_{fair}}{r_i})_+$, which limits each flow's throughput to the fair share r_{fair}. Thus, dropping is not applied uniformly across flows but is applied differentially based on an estimate of a flow's current sending rate. The fair rate r_{fair} is estimated implicitly based on the length of the FIFO queue, and so requires little state or complexity. Providing reasonably accurate estimates of the flow rates r_i is the key technical challenge.

The state required to accurately estimate each flow's rate is quite large, growing linearly with the number of flows, and thus is infeasible. However, we only need to keep state for those flows that are sending at or above the fair share, since those are the only flows whose packets will be dropped. Recent trace data has suggested that the distribution of flow rates is long-tailed and that most bytes are sent by relatively *fast* flows. For example, Figure 1 shows the cumulative distributions of the 1-second flow rates for three traces; in these datasets, 10% of the flows represent between 60% to 90% of the total bytes. Thus, a sampling of the recent traffic will be heavily dominated by the faster flows and, in typical cases, these are the only flows at or above the fair share. Our design uses a sampling of recently arrived packets to estimate the flow rates. The state required is roughly proportional to the number of *fast* flows, not the total number of flows, and thus is much more manageable. Back-of-the-envelope calculations indicate that this state will be much less than that already kept in the packet buffers.

A direct implementation of AFD algorithm, which we refer to as AFD-SB, requires two data structures: a shadow buffer which stores the recent history of all packet arrivals (header only) and a flow table which keeps the packet count of each flow that appeared in the shadow buffer. A randomized approximation of AFD, which we call AFD-FT, uses only a flow table and, as a result, it needs much less state than AFD-SB.

We have evaluated AFD in a variety of scenarios using simulations. One typical simulation is presented in Figures 2 and 3, in which 7 TCP flow groups (5 flows each) with different congestion control mechanisms and RTTs compete for a congested link bandwidth of 10Mbps. Figure 2 shows the average throughput that each flow group gets under four algorithms: RED, FRED, AFD-SB and AFD-FT. The corresponding drop probability of each flow group is depicted in Figure 3. This and other data suggests that, in a wide range scenarios, AFD provides a good approximation to fair bandwidth allocation, typically providing bandwidth allocations within +/-15% of the fair share.

[1] A recent paper describes the RED-PD algorithm, which has similar but not identical goals to what we propose here.