Forty-Sixth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
September 23-26, 2008

Approximate fair bandwidth allocation: A method for simple and flexible traffic management

Rong Pan, Balaji Prabhakar, Flavio Bonomi and Bob Olsen

Abstract— **The Internet architecture uses congestion avoidance mechanisms implemented in the transport layer protocol like TCP to provide good service under heavy load. If network nodes distribute bandwidth fairly, the Internet would be more robust and accommodate a wide variety of applications. Various congestion and bandwidth management schemes have been proposed for this purpose and can be classified into two broad categories: Packet scheduling algorithms such as Fair Queueing (FQ [7]) which explicitly provide bandwidth shares by scheduling packets. They are more difficult to implement compared to FIFO queueing. The second category has active queue management schemes such as RED [10] which use FIFO queues at the routers. They are easy to implement but don't aim to provide (and, in the presence of non-congestion-responsive sources, don't provide) fairness. An algorithm called AFD (Approximate Fair Dropping) [18]), has been proposed to provide approximate, weighted max-min fair bandwidth allocations with relatively low complexity. AFD has since been widely adopted by the industry. This paper describes the evolution of AFD from a research project into an industry setting, focusing on the changes it has undergone in the process. AFD now serves as a traffic management module, which can be implemented either using a single FIFO or overlaid on top of extant per-flow queueing structures and which provides approximate bandwidth allocation in a simple fashion.**

The AFD algorithm has been implemented in several switch and router platforms at Cisco Sytems, successfully transitioning from the academic world into the industry.

I. Introduction

Note: This paper describes the background, design and commercial deployment of the bandwidth allocation scheme AFD (Approximate Fair Dropping) which was developed in [18] and [17]. Our goal is to describe the ultimate design decisions that were made to enable its commercial implementation. We refer the interested reader to [18] and [17] for the motivation, ideas and original choices behind AFD, where Lee Breslau of AT&T and Scott Shenker of UC Berkeley are crucial contributors. This paper should be viewed as a continuation of the work on AFD and a chronicle of its evolution from an academic research subject into the real world of routers and switches.

Background. We briefly review some necessary background. By design, the Internet delivers a connectionless, best-effort type of service. With ubiquitous connectivity as its primary goal, its reliable functioning is based on the premise that end hosts shall be responsible for the correct and complete exchange of data, leaving intermediate network nodes to facilitate data forwarding. Indeed, with end hosts employing transport layer protocols, like TCP (Transmission Control Protocol) to regulate the amount of work injected into the network, network routers are freed up to route and forward data at high speeds to their final destinations. This infrastructure has allowed the Internet to grow at an astonishingly fast rate.

The introduction of many new applications has brought a new mix of traffic into the Internet with different requirements of bandwidth, loss and latency. In addition, service differentiation has led network equipment vendors to develop products for different market segments; notably, enterprise, metro, edge and core networks. This has resulted in network nodes (switches, routers) taking a more active role in managing their resources, notably bandwidth management and allocation.

There has been a lot of research on developing mechanisms for fairly partitioning the bandwidth at a link among the flows traversing it, resulting in algorithms such as FQ [7], CSFQ [21], SFQ [14] and RED [10], FRED [12], SRED [16], SFB [9], RED-PD [13]. These schemes represent a spectrum with high-precision bandwidth allocation on one end (typified by the Fair Queueing, or FQ, algorithm) and simple congestion management on the other end (typified by the Random Early Detection, or RED, algorithm). The cost of implementation varies inversely as the quality of bandwith-allocation and FQ is much more expensive to implement than RED.

Different choices along this spectrum of router algorithms embody different expectations about the set of congestion algorithms employed at end hosts. One possible view is that in the future almost all flows will use a TCP-compatible congestion control algorithm, and that there will only be a very few malicious (or broken) flows that are substantially more aggressive. In this scenario, routers only need to detect these aggressive flows and restrain their bandwidth usage. The CHOKe algorithm [19] is an example well-suited to this task. The amount of extra work it performs is of the order of non-TCP-compatible (or aggressive) flows. Another possible view is that flows will use a very wide variety of congestion control algorithms, some of which not necessarily TCP-compatible, and that routers will need to perform some degree of bandwidth allocation. The AFD (Approximate Fair Dropping) algorithm is designed for this scenario. Based on the observation that Internet flow distribution is heavy-tailed [18] (most flows are small and low-rate 'mice', and a few large 'elephant' flows bring the most work), its state requirement is of the order of the number of large elephant flows. We refer the reader to [18] for details.

Rong Pan, Flavio Bonomi and Bob Olsen are with Cisco systems, San Jose, California. Balaji Prabhakar is with the Department of Electrical Engineering, Stanford University, `balaji@stanford.edu`.

Bandwidth partitioning in the industry. A significant amount of research and development on badwidth partitioning occurred in the industry, originating in ATM technology under the practice of traffic management. Traffic management algorithms aim to achieve very precise allocation and control of bandwidth, requiring that precise scheduling, shaping and policing decisions be made at the time scale of an ATM cell (a 53-byte packet). Scheduling algorithms such as Weighted Fair Queuing (WFQ) when combined with shaping and policing mechanisms such as the Leaky Bucket provide a high precision of bandwidth partitioning and a tight control of end-to-end latency (see [2] for details. But, such precise bandwidth scheduling algorithms come at considerable expense, and by abandoning precise bit-level fairness in favor of quantum-level[1] fairness, has enabled the Deficit Round Robin (DRR) [20] scheme to be widely-deployed in routers and switches.

But DRR still requires per-flow queues and an *a priori* commitment of hardware resources, based on the number of flows that will be supported. This is both expensive and inflexible: it is not possible to add a single flow in excess of the number for which queues have been provided.

Approximate bandwidth partioning. The AFD algorithm takes a further step down the path of approximate bandwidth partitioning: it aims to provide bandwidth at the level of a few tens of milliseconds (equivalent to a few hundred packets). Such a trade-off in precision is supported by the needs of network traffic; bandwidth measured at the level of tens of milliseconds is adequate for voice, video and data. The AFD algorithm is founded on two fundamental ideas: (i) per-flow queues are not necessary for partitioning; per-flow monitoring of rates, which is "soft per-flow state," is adequate and (ii) bandwidth partitioning can be achieved by dropping packets from a flow which exceeds its allocated rate, rather than by explicitly providing it the correct rate using packet scheduling algorithms. These ideas at once simplify the implementation and allow for AFD to be an enhancement to more fine-grained packet scheduling algorithms, such as DRR. Thus, implementations can overlay AFD on top of DRR to achieve various degrees of precision in bandwidth partitioning while becoming more flexible (by not having a hard upperbound on the number of flows that can be supported) and affordable.

Since AFD aims for approximate fairness over long time scales, of the order of several roundtrip times, its design is easily amenable to high-speed implementation. AFD performs probabilistic drop-on-arrival. When a packet arrives, it is either dropped or placed on a physical queue. The drop decision is simple with O(1) complexity (that is, the complexity does not increase with the number of classes/flows or packets).

AFD has many applications in the area of traffic management. Depending on the capability of the underlying scheduling and queuing system, AFD offers the ability to provide a richer set of functionalities. For example, AFD may be used as a replacement for the Weighted RED algorithm. Also, for systems that lack adequate physical queues, AFD provides the ability to support a behavior consistent with a system that has a larger number of queues. Similarly, for systems lacking sufficient layers of hierarchy, AFD can provide this enhancement. In summary, AFD offers the potential to effectively complement, extend and simplify a broad range of current traffic management mechanisms.

[1]A quantum of data is typically 500 Bytes.

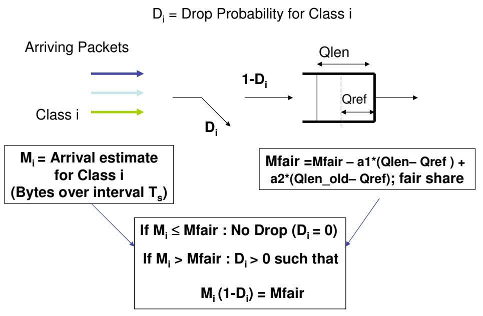
D_i = Drop Probability for Class i
Arriving Packets
Class i
Qlen
$1\text{-}D_i$
Qref
D_i
M_i = Arrival estimate for Class i (Bytes over interval T_s)
Mfair =Mfair – a1*(Qlen– Qref) + a2*(Qlen_old– Qref); fair share
If $M_i \le$ Mfair : No Drop ($D_i = 0$)
If $M_i >$ Mfair : $D_i > 0$ such that
$M_i(1\text{-}D_i)$ = Mfair

Fig. 1. Basic AFD Algorithm

The remainder of this document is structured as follows: we describe the details of AFD algorithm in Section II and we introduce a few of AFD's applications and their performance in Section III. We conclude in Section IV.

II. AFD Algorithm: Details

The fundamental goal of the AFD algorithm is to control bandwidth allocation among flows/classes that share a common queuing system. Controlled bandwidth allocation is achieved by selectively, probabilistically dropping the packets of classes that send more than their "fair share," which depends on the notion of fairness we adopt.

A. Basic AFD Algorithm

As mentioned earlier, AFD is an active queue management scheme that drops packets probabilistically upon arrival. A key aspect of AFD is that the decision of whether to drop a packet from a given class i is based not only on some measurement of queue depth, but also on an estimate of the class i's current sending rate r_i. This is different from other active queue management systems (e.g., RED) where the decision is based solely on the queue depth.

Let $D_i = (1 - rfair_i/r_i)_+$ denote the probability with which a packet from class i should be dropped. Thus, if $r_i < rfair_i$ no drop will occur. If $r_i > rfair_i$, the drop probability increases as r_i gets further away from $rfair_i$. As a result, the throughput of each flow, $r_i(1 - D_i)$, is bounded by its fair share: $r_i(1-D_i) = min(r_i, rfair_i)$. Hence, drops do not occur evenly across flows but are applied differentially to flows with different rates.

Two key algorithmic aspects of AFD lie in the manner in which r_i and $rfair_i$ are estimated via measurements. There are three elements in this procedure, as shown in Figure 1:

1. Arrival Rate Estimation. Let $M_i = r_i T_s$ be the amount (measured in bytes, packets, etc) of traffic from flow

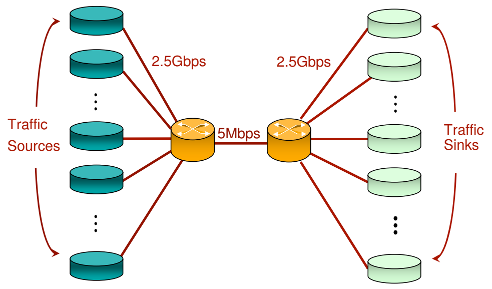

Fig. 2. Simulation Topology

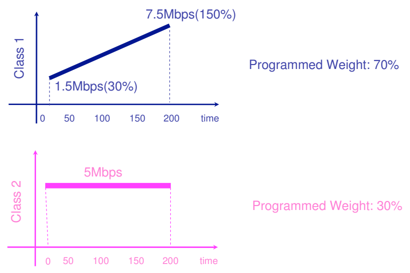

Fig. 3. AFD vs. WRED: Traffic Scenario

i during the interval T_s. Similarly, $Mfair$ be the fair share amount of the traffic that the queue would have received if sources sent at their fair share rates $Mfair_i$, which will be estimated as described next.

2. Fair Share Calculation. The fair share of the link bandwidth is estimated dynamically, at the end of each measurement interval, as follows. First, use the queue-size to dynamically estimate $Mfair$ using the equation: $Mfair = Mfair - a_1 * (Qlen - Qref) + a_2 * (Qlen_{old} - Qref)$, where $Qlen$ is the real, instantaneous, queue length measured at the end of the current measurement interval; $Qlen_{old}$ is the real queue length measured at the end of the previous interval; Q_{ref} is the reference queue length (set by the operator); a_1 and a_2 are the averaging parameters (chosen as part of the design); and $Mfair_i = w_i * Mfair$, where w_i is the weight associated with class i.

3. Drop Probability Calculation. When a flow i packet arrives, we use a drop probability D_i which satisfies

$$M_i * (1 - D_i)_+ = min(M_i, Mfair_i).$$

From the equations presented in this section, it is clear that AFD is a closed loop control system. A detailed stability analysis of this system has been developed but not presented here. Such stability analysis is important for determining stability margins and in correctly chosing the parameters of the algorithm.

III. AFD Applications

We will review some applications of AFD as an active queue management scheme which improves on WRED and which can closely emulate the bandwidth allocation achieved by more complex queuing-plus-scheduling schemes such as Weighted Deficit Round Robin (WDRR) [20]. We also show how AFD may be extended to cover a number of new applications related to traffic management; these applications demonstrate the flexibility and generality of AFD. We use Network Simulator, ns [7], for our simulations.

Consider the simulation topology depicted in Figure 2. The access link speed is 2.5Gbps and the bottleneck link has a bandwidth of 5Mbps.

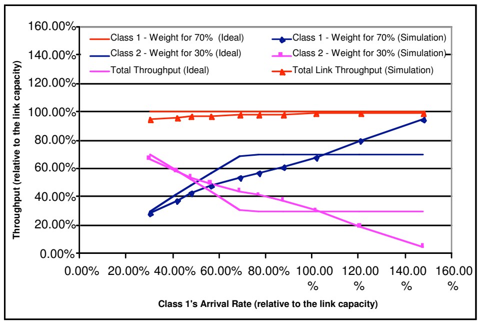

Fig. 4. WRED vs. Ideal

A. AFD vs Weighted RED

WRED [6] is a variation of the RED algorithm which aims to differentially drop the packets of different classes, thereby prioritizing the classes. Consider the performance of providing differentiated, or weighted, allocation of bandwidth by using either the WRED algorithm or the weighted-AFD algorithm. We consider an experiment including two traffic classes, competing for a link of capacity C = 5 Mbps, with a desired bandwidth split of 70% to 30% between class 1 and class 2. Class 2 gradually increases its sending rate from 30%C to 150%C, while Class 1 sends rate C all the time, as shown in Figure 3.

The bandwidth allocation achieved by WRED is plotted against the ideal performance in Figure 4. The WRED parameters are chosen in such a way that they provide the desired bandwidth partitioning when class 2 sends at 100% of the link capacity C. However, when the total traffic intensity is different from this load, it is clear from the plot that WRED is not able to allocate bandwidth in the desired proportion. As the overall traffic intensity increases, the aggressive flow, Class 1 in this case, grabs more bandwidth virtually shutting out Class 2.

By contrast, AFD using weights 7 and 3 for Classes 1 and 2 respectively, achieves an almost perfect behavior as shown in Figure 5. Initially, with its demand less than its allocated rate, Class 1 gets its requested bandwidth and Class 2 obtains the remaining bandwidth. When Class 1 offers more than 70% of the link bandwidth, Class 1's throughput is restricted

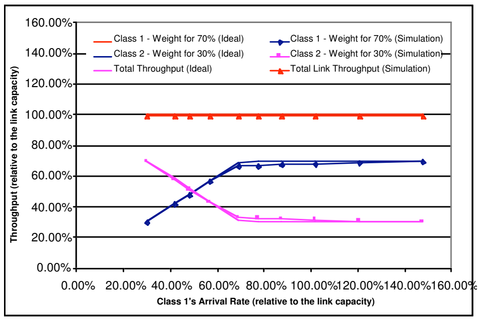

Fig. 5. AFD vs. Ideal

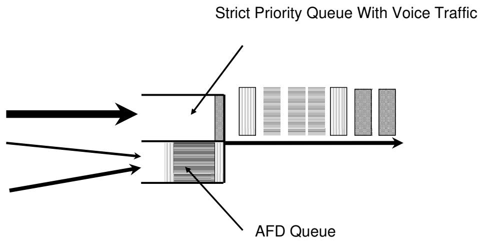

Fig. 6. AFD vs WDRR: Simulation Setup

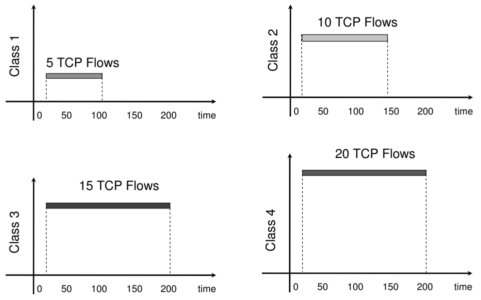

Fig. 7. AFD vs WDRR: Four Classes of TCP Traffic

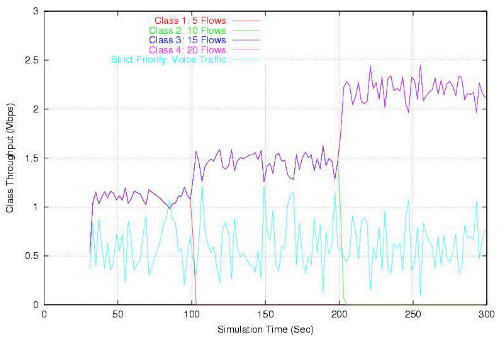

Fig. 8. Performance Under WDRR

to 70% of the line rate and Class 2 gets the remaining 30%, as desired.

B. AFD vs. WDRR

This section discusses how AFD can be used to complement a more traditional queuing and scheduling system, while replacing certain components of the more costly queueing and scheduling system.

We study the behavior of AFD when the scheduler of the FIFO queue does not have a constant throughput. We introduce a strict priority queue into our system as shown in Figure 6. Under this scenario, the capacity on the non-strict priority is effectively varying as it can only take the left-over bandwidth unused by the strict priority queue. We need to make sure that once again fairness is enforced under AFD with varying serving speed of the FIFO queue.

An exponential on-off traffic source with an average load of 12%C is used to represent the high-priority voice traffic. For the lower priority traffic, we have four classes of TCP as follows: 5 flows in Class 1, 10 flows in Class 2, 15 flows in Class 3 and 20 flows in Class 4. Thus, the sending rate in each class is different. The classes have an equal weight of 1 and the flows are on and off at different times as shown in Figure 6; that is, all classes are on at the beginning of the simulation, then Class 1 stops sending traffic at 100 second and Class 2 finishes transmitting at 150 second; Classes 3 and 4 send traffic throughout the simulation.

Once the voice traffic obtains its bandwidth by virtue of belonging to the strict prioty queue, the four TCP traffic classes should share the remaining link bandwidth equally. Since the fair share for these four classes is varying, we use DRR as our reference to evaluate the performance of AFD. Figure 8 depicts the throughput of each class under DRR, and it is clear from the plot that each of the four TCP classes obtains equal bandwidth.

Figure 9 shows that the performance of AFD matches that provided by DRR quite closes. Indeed, a closer look of the throughputs of Class 4 under the two different schemes (Figure 10 shows that AFD is essentially identical to DRR in performance.

IV. Conclusion

In this paper, we have shown that AFD provides a rich set of functionalities which is complementary to the existing queue/scheduling structures. AFD also can enhance the capabilities of systems that currently don't support some advance features such as service propagation and dynamic policing. Furthermore, AFD offers the capability of supporting non-linear excess sharing rules that is difficult to provide in traditional systems.

The AFD algorithm has been used in several switch and router platforms at Cisco Systems, making the transition from an academic project into wide use in the industry.

References

[1] Bansal, D., and Balakrishnan, H., "Binomial Congestion Control Algorithms " *Proceedings of Infocom '2001*, April 2001.
[2] Bertsekas, D., and Gallager, R., "Data Networks", December 1991.
[3] `http://www.caida.org/analysis/AIX/plen_hist/`

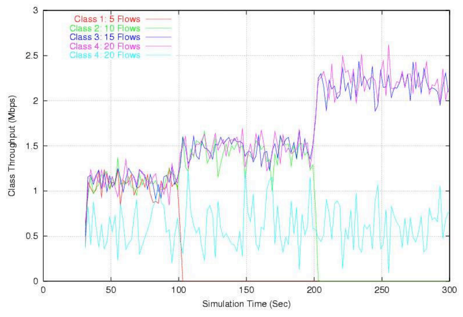

Fig. 9. Performance Under AFD

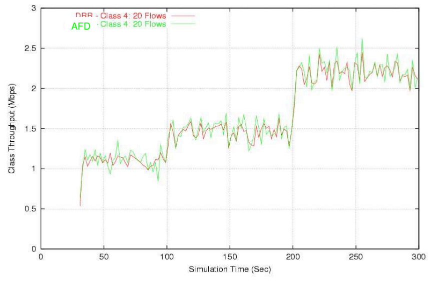

Fig. 10. AFD vs. WDRR

[4] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S.,Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L.,Ramakrishnan,K., Shenker,S., Wroclawski, J., Zhang, L., "Recommendations on queue management and congestion avoidance in the internet", *IETF RFC (Informational) 2309*, April 1998.

[5] Briscoe, B., "Flow Rate Fairness: Dismantling a Religion", *ACM SIGCOMM Computer Communication Review*, April 2007.

[6] Cisco White Paper, `http://www.cisco.com/warp/public/cc/pd/iosw/ioft/ioqo/tech/qos_wp.htm/`

[7] Demers, A., Keshav, S. and Shenker, S., "Analysis and simulation of a fair queueing algorithm", *Journal of Internetworking Research and Experience*, pp 3-26, Oct. 1990. Also in Proceedings of ACM SIGCOMM'89, pp 3-12.

[8] Evans, J., Filsfils, C., "Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice", *Morgan Kaufmann*, 2007.

[9] Feng, W., Shin, K., Kandlur, D. and Saha, D., "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness", *Proceedings of INFOCOM'2001*, April, 2001.

[10] Floyd, S. and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transaction on Networking*, 1(4), pp 397-413, Aug. 1993.

[11] Floyd, S., and Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, August 1999.

[12] Lin, D. and Morris, R., "Dynamics of random early detection", *Proceedings of ACM SIGCOMM'97*, pp 127-137, Oct. 1997.

[13] Mahajan, R. and Floyd, S. "Controlling High-Bandwidth Flows at the Congested Router", ACIRI, Berkeley, California, Nov. 2000.

[14] McKenny, P., "Stochastic Fairness Queueing", *Proceedings of INFOCOM'90*, pp 733-740.

[15] Nagle, J., "On packet switches with infinite storage", *Internet Engineering Task Force*, RFC-970, December, 1985.

[16] Ott, T., Lakshman, T. and Wong, L., "SRED: Stabilized RED", *Proceedings of INFOCOM'99*, pp 1346-1355, March 1999.

[17] Pan, R., Breslau, L., Prabhakar, B. and Shenker, S., "Approximate Fair Allocation of Link Bandwidth", *Hot Interconnect 2002*, also appeared at *IEEE Micro*, January 2003.

[18] Pan, R., Breslau, L., Prabhakar, B. and Shenker, S., "Approximate Fairness Through Differential Dropping", *Computer Communication Review*, January 2003.

[19] Pan, R., Prabhakar, B. and Psounis, K., "CHOKe - A Stateless Active Queue Management Scheme For Approximating Fair Bandwidth Allocation", *Proceedings of INFOCOM'00* March 2000.

[20] Shreedhar, M., and Varghese, G., "Efficient Fair Queueing using Deficit Round Robin", *ACM Computer Communication Review*, October 1995.

[21] Stoica, I., Shenker, S. and Zhang, H., "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks", *Proceedings of ACM SIGCOMM'98.*

[22] Xu, Y., Guerin, R., "On the Robustness of Router-based Denial-of-Service (DoS) Defense Systems", *ACM SIGCOMM Computer Communication Review*, July 2005.

[23] ns - Network Simulator (Version 2.1b26).