A New Service Differentiation Scheme: Size Based Treatment

Stylianos Dimitriou, Vassilis Tsaoussidis
Dept. of Electrical and Computer Engineering, Democritus University, Greece
{sdimitr, vtsaousi}@ee.duth.gr

Abstract

The increase of real–time applications as well as the vast usage of portable wireless devices has led to a corresponding increase of real–time traffic with strict bandwidth and delay requirements. In order to satisfy the requirements of various applications we typically use Service Differentiation. However, as new real–time applications are created, their portion of the total traffic increases, thus making more and more difficult to satisfy completely their requirements. In this paper we propose a new scheme, which is based on the axiom that 'different types of applications typically utilize different packet sizes'. With Size Based Treatment (SBT) different packet sizes are dropped with different probability by the queue. Small sized packets can benefit and transmit on higher rates, increasing the total system fairness.

Keywords – Active Queue Management, Dropping Algorithms, Fairness, Service Differentiation

1. Introduction

In the early years of the Internet, data transmission involved either bulk data transfer (FTP) or small data transfer with minor time constraints (HTTP, SMTP, Telnet). Today, with the introduction of streaming technologies, we have applications that require timely data delivery and low loss ratio. The demand for real–time data has increased not only because of these emerging technologies, by also because there are a lot of portable wireless devices that connect to the Internet and support multimedia content, such as new generation mobile phones and PDAs. Applications like VoIP and instant messaging demand fast and lossless data delivery. On the other side, bulk data transfer is the biggest part of Internet traffic, not only due to FTP, but also due to new protocols such as BitTorrent [1] that facilitate big data transfer among peer stations. The need for Service Differentiation is more emerging today than the past years. The DiffServ [2] architecture defined a general framework for service differentiation, which included satisfying applications needs for throughput, delay, jitter and packet loss. Applications with different characteristics have different requirements and their packets should be treated differently. In DiffServ, packets are marked to receive different – per hop – treatment, thus we need to mark each packet with the applications requirements. However, we know that most real–time applications and protocols use small packet sizes (ATM uses 53 byte packets and VoIP 60–170 byte packets). Moreover, protocols that deal with messaging and signaling, such as ICMP, use small packet sizes. Furthermore the majority of control packets (SYN, FIN, ACK) for TCP and other protocols are also small packets and vary from 20 to 40 bytes. Their timely delivery is critical and multiple losses result occasionally in severe quality degradation. Due to their small size, they do not contribute significantly, in general, to network contention and they usually have constant sending rates. Applications that use FTP and BitTorrent use packets whose size is bigger than 1KB. Their payload to overhead ratio is high and we can tolerate small losses and out–of–order delivery, since they are not delay–sensitive. From now on, we will refer to flows that generate small packet sizes as 'small flows' and to flows that generate big packet sizes as 'big flows'. This discrimination has nothing to do with the duration of the connections or the total amount of data transmitted, only with the size of packets generated by the flow.

In this paper we propose a technique to differentiate service based on packet size. However, setting a single or multiple static thresholds and treating differently the packets that fall among these thresholds would result in undesirable behavior. The binary classification among 'big' and 'small' packets should be relative to the size of packets that already have been served by the router. That means that a 500 byte packet is considered small from a router that recently served 1KB packets and big from a router that recently served 100 byte packets. Moreover, our ability to satisfy the bandwidth needs of

978-1-4244-2036-0/08/$25.00 ©2008 IEEE

different packets varies as the proportion of small and big packets in a buffer changes. A buffer almost full of big packets should be able to accommodate and serve a small packet; however, as more small packets arrive, less small packets should be favored at the expense of the bigger packets. Another point to consider is that in real life, differentiation is not binary; we cannot consider only small and big packets. Both 100 and 800 byte packets are smaller than a 1000 byte packet, yet we should not favor equally these two packets over the bigger one. Last, it is important to create an algorithm that would not affect end–nodes and prevent misbehaving flows from claiming more bandwidth than their fair share.

A packet dropping method appears appropriate to implement a size based algorithm. We deploy a new paradigm by modifying RED's dropping algorithm in a manner that favors small packets over bigger. Size Based Treatment will cause drops less frequently for packets that are smaller by the average size of the packets served by the queue. The dropping probability will be higher for packets that are close to the average, but smaller than packets that are much bigger than the average. In our experiments we evaluate SBT's and demonstrate its ability to significantly increase fairness among different packet sizes by prioritizing small packets. We show that smaller packets have more opportunities to survive in high contention environments than before, causing minor impact on the bigger packets.

Section 2 presents the work that has been done over service differentiation schemes. In section 3 we describe the desirable behavior of a service differentiation scheme and present our proposal. In section 4 we evaluate experimentally SBT and in section 5 we conclude and set the framework for future work.

2. Related Work

Service differentiation has been developed on the basis of resource reallocation in order to satisfy the requirements of applications with different characteristics. Bandwidth and buffer space are distributed based on each applications specific needs. The need for service differentiation has arised from the increasing demand for real–time applications, like multimedia streaming and IP telephony. These applications have strict delay, jitter and loss constraints which can be satisfied usually by prioritizing real–time data over bulk data transfers.

Although the DiffServ architecture proposes marking a packet with the service it should receive, many proposals have been made in the basis of modifying dropping and scheduling mechanisms of the router in order to support service differentiation without affecting the end–nodes. In [7], Floyd and Fall introduced mechanisms based on the identification of high bandwidth flows from the drop–history of RED. In [4] the authors propose an explicit allocation of bandwidth to various flows based on their respective needs and determine this allocation by modifying accordingly the dropping probability. Weighted RED with Thresholds (WDT) [3] calculates a separate average length for the higher–priority packets, preventing starvation for the lower–priority traffic. Flow RED (FRED) [8] uses per–active–flow accounting to impose on each flow a loss rate that depends on the flow's buffer use. Unfortunately, extended memory and processor power is required for a big number of flows. On the other hand RED–PD (Preferential Dropping) [9] maintains a state only for the high–bandwidth flows and drops their packets more frequently than packets from low–bandwidth flows.

Fair Queuing [5] maintains equal queues for each flow and in Weighted Fair Queuing the queues can have different length. Core–Stateless Fair Queuing [13] uses two types of routers; edge and core. Edge routers compute per–flow rate estimates and label the packets with these estimates, whereas core routers drop the packets probabilistically based on these labels. The CHOKe mechanism [11] tries to identify flows that occupy highly the queue by matching every incoming packet against a random packet in the queue and either drop both, if they belong to the same flow, or accept them with a certain probability. Stochastic Fair BLUE [6] and ERUF [12] aim to identify and rate–limit non–responsive flows. Last, NCQ [10] distinguishes data into congestive and non–congestive queuing (minimal–size packets) and favours non–congestive packets over congestive during scheduling. We depart from the same idea, however we emphasize on the dropping strategy of the router.

3. Size Based Treatment

Small packets require small transmission times, occupy less buffer space and may receive better service against bigger packets. In cases of maximum queue utilization this advantage becomes more significant since it is more probable that a bigger packet overflows the queue.

In Figure 1, we can see that, in general, small packets are favored by DropTail's dropping, as they participate less in buffer overflows.

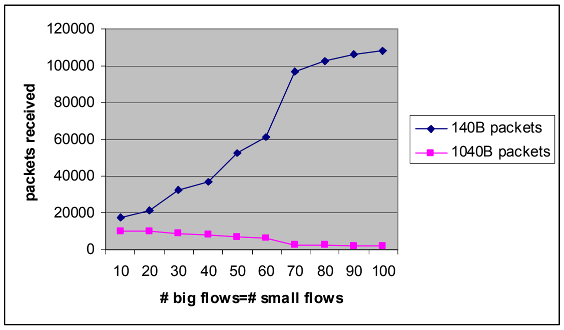


Figure 1. 140B vs 1040B packets in a DropTail queue

However, RED's proactive dropping minimizes buffer overflows and the aforementioned advantage becomes less significant. In RED all packets are treated the same; they are dropped with the same probability regardless of their size. Even though the number of packets sent may be bigger for the small flows, the actual Throughput will be less. Eventually, the smaller flows will starve for resources.

Before presenting our proposal, we should outline the desirable behavior for SBT.

We highlight the following scenarios;

1. The queue is occupied only by big packets; a much smaller packet arrives. In this case we want to be sure that the small packet has maximum priority over big packets and would not be dropped. As the number of small packets increases, we should increase the dropping probability of smaller packets, since they now consist a significant proportion of the router's traffic.

2. The queue is occupied only by small packets; a big packet arrives. In this case, since traffic consists mostly from small packets, we want the big packet to have similar treatment as the rest. If more big packets arrive, the queue should gradually decrease small packets dropping probability and serve them better over big packets.

3. The queue is occupied by big packets; a medium packet arrives. The packet in this case would have better treatment than the rest, nevertheless worse compared to the service a much smaller packet would receive from the router.

Furthermore, we would like to make sure that we avoid misbehaving flows;

1. The avg_size is not known a priori. Fragmenting the packets to smaller segments may result in increased overhead. A 1000 byte packet has 40 bytes (4%) overhead, whereas a fragmented 1000 byte packet into 100 byte packets will have 400 bytes (40%) overhead. In this way, any benefits on Throughput from fragmenting would be canceled by the increased overhead.

2. SBT is based on the fact that packets may have any size. As more flows start to misbehave, the avg_size will lower and the prioritizing scheme will become obsolete. The same increased overhead result will affect all flows. A flow that will not misbehave will achieve the same service as misbehaving flows with less overhead.

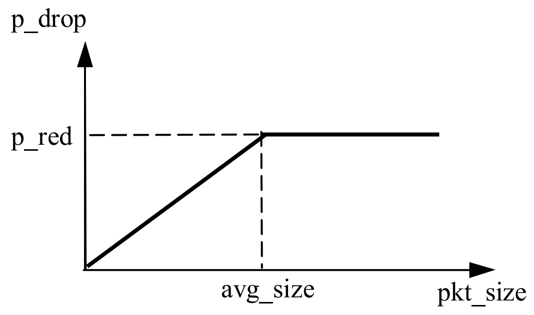


Figure 2. SBT's dropping probability

Based on these remarks we form SBT's algorithm. We consider an avg_size, which is a moving average of the incoming packet sizes. Packets whose size is bigger than avg_size will be treated as usual. Packets which are smaller than avg_size will be dropped with smaller probability, depending on their difference from avg_size. P_red indicates the dropping probability calculated by the original RED algorithm, and p_drop indicates the dropping probability of SBT. We have,

$$p\_drop = p\_red \frac{pkt\_size}{avg\_size}, \text{ if } pkt\_size < avg\_size$$

$$p\_drop = p\_red, \text{ if } pkt\_size \geq avg\_size \text{ and}$$

$avg\_size = \alpha \cdot pkt\_size + (1 - \alpha) \cdot avg\_size$, where α takes small values usually between 0.1 and 0.2.

As we can deduce from Figure 2 and from the formula above, the dropping probability for a small packet will be determined not only by its size, but also by the difference of its size from avg_size. A buffer occupied with equal number of small packets (140 bytes) and big packets (1040 bytes) will serve a 540 byte packet with almost no special priority, even though there are bigger packets in the buffer.

4. Simulations

We demonstrate SBT's ability to achieve better fairness among packets of different sizes, and consequently services. We consider a simple dumbbell topology with varying proportions of small and big packets (Figure 3).

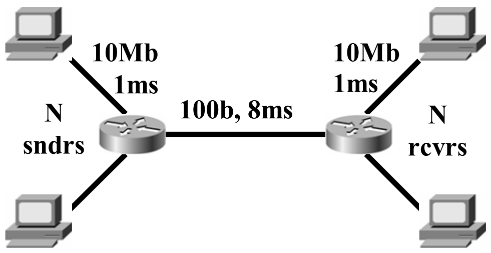
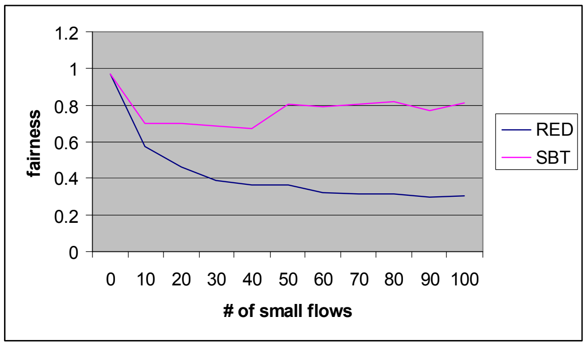


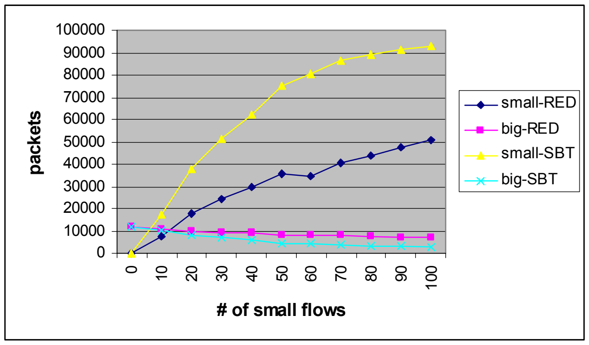
Figure 3. The Dumbbell Topology used

For the first set of simulations we use 140 and 1040 byte packets and for the second set 540 and 1040 byte packets. We first consider 10 big flows and we increase the number of small flows, then we consider 10 small flows and we increase the number of big flows and last we consider equal numbers of small and big flows. Both small and big flows use TCP Newreno. The simulations where done in ns–2.

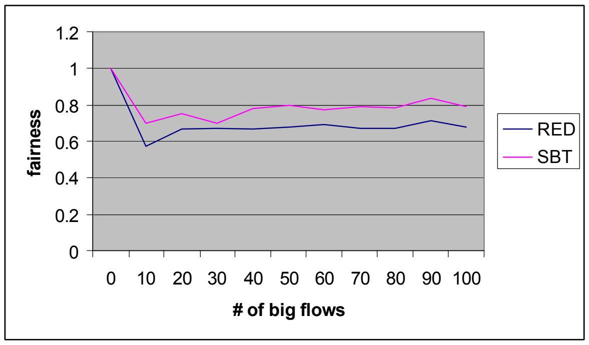
4.1. Small flows: 140B packets

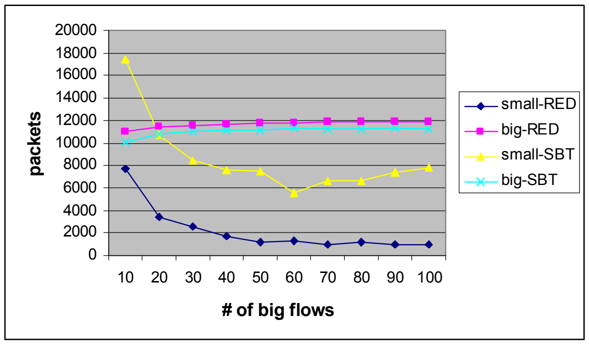
In this first case we consider 140 byte small packets and 1040 byte big packets. We observe a significant increase in fairness for each of the three scenarios, especially for the first scenario, where 'small' flows are more than 'big' flows and can achieve fair bandwidth share. In the second scenario, the fairness increase is smaller, yet as we can see from Fig. 7, even when we have 100 big flows and 10 small flows, small flows will not starve. The third scenario is the most interesting, because we see that with SBT, small flows can increase dramatically their throughput as more small flows participate in contention, whereas with RED, they were let to starve.



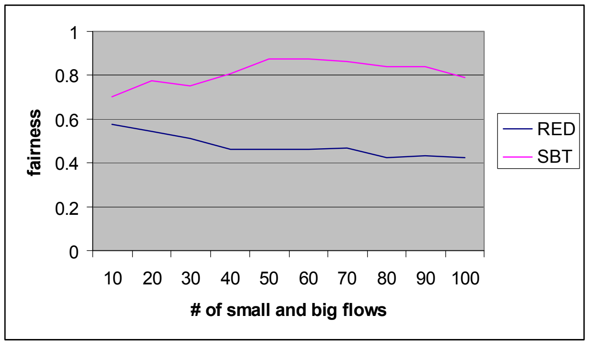


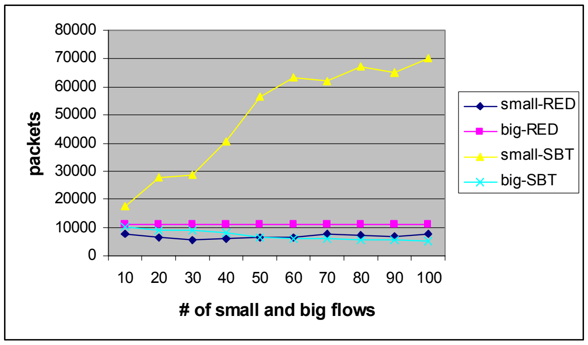
Figures 4,5. 10 big flows

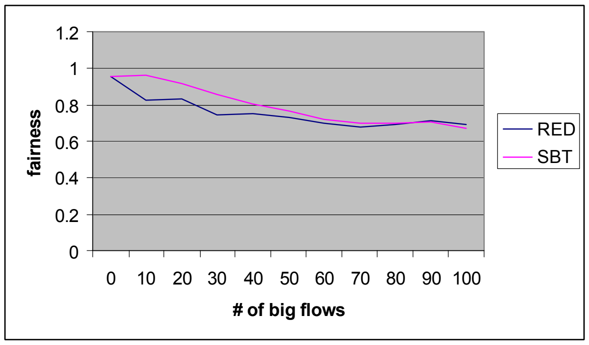




Figures 6,7. 10 small flows



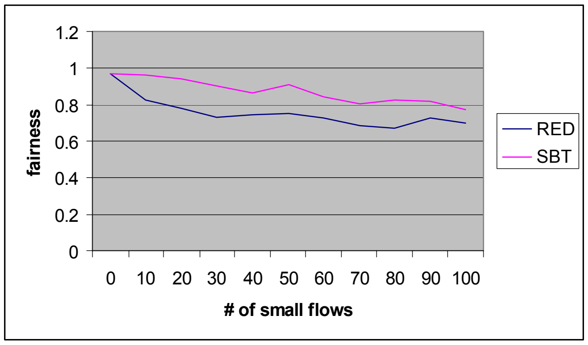


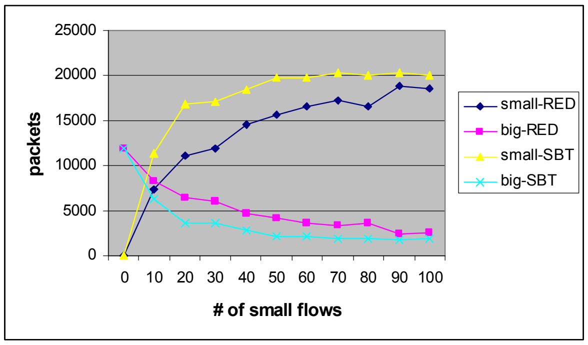


Figures 8,9. Equal number of small flows and big flows

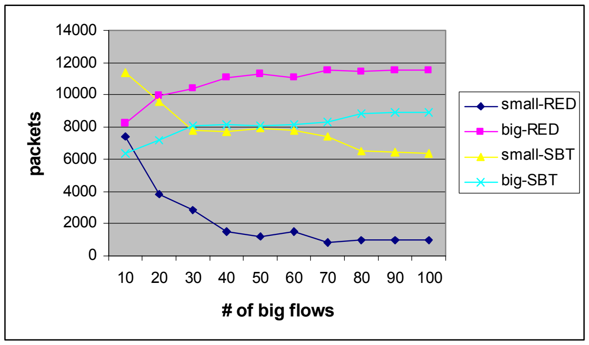
4.2. Small flows: 540B packets

The results we get from the second case are not as significant as before, indicating that SBT prioritizes packets depending on how smaller the packets are from the average size, not only by they fact that they are smaller. However there is an interesting point; Figure 12 shows that fairness is decreased. This is caused by the increased priority small flows have over bigger, to the point that the router becomes "unfair" over big flows. At the first scenario we observe that small packets have no special treatment over big. Many small packets cause a small avg_size, which in turn gives them almost the same dropping probability as big packets, thus the difference on performance is minor, yet capable of an almost 15% increase on fairness.

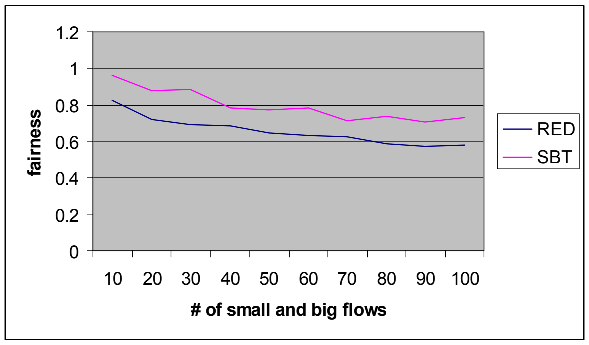


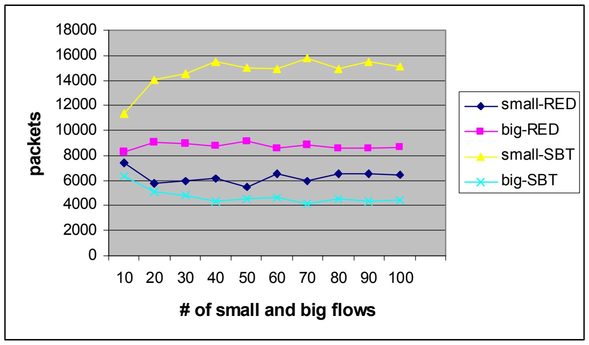


Figures 10,11. 10 big flows



Figures 12,13. 10 small flows





Figures 14,15. Equal number of small and big flows

5. Conclusion and future work

We have proposed a new Service Differentiation technique that corresponds service to packet size. The first results are encouraging since we could significantly increase the system fairness and prioritize small over big packets with minor cost for the latter. Unlike other similar proposals, SBT is easily deployable, since it doesn't require end–user modifications, and uses only one state variable for each router. In the future we would extend this work to more complex topologies which would involve multiple routers in a row. In would be interesting to see if big TCP flows and small UDP (constant rate) flows would result in link underutilization. Additionally, we could examine if there are other characteristics for a packet, apart from size, that are easily extracted without much computational cost, like the ACK flag. These would help us classify more accurately the incomings packet and serve them more fairly than now. Finally, an analytical result for evaluating the performance of SBT is also under development.

6. References

[1] BitTorrent, http://www.bittorent.com

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "RFC2475 – An Architecture for Differentiated Services", December 1998

[3] U. Bodin, O. Schelen, and S. Pink, "Load–tolerant Differentiation with Active Queue Management", *CCR*, July, 2000

[4] D.D. Clark and W. Fang, "Explicit Allocation of Best–Effort Packet Delivery Service", *IEEE/ACM Transactions on Networking*, August 1998

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", *Proceedings of SIGCOMM 1989*, September 1989

[6] W.C. Feng, D.D. Kandlur, D. Saha, and K.G. Shin, "Stochastic Fair Blue: A Queue Management for Enforcing Fairness", *INFOCOM 2001*, April 2001

[7] S. Floyd and K. Fall, "Promoting the use of end–to–end congestion control in the Internet", *IEEE/ACM Transactions on Networking*, May 1999

[8] D. Lin, and R. Morris, "Dynamics of Random Early Detection", *SIGCOMM 1997*, September 1997

[9] R. Mahajan, and S. Floyd, "Controlling High Bandwidth Flows at the Congested Router", *ICNP 2001*, November 2001

[10] L. Mamatas, and V. Tsaoussidis, "A new approach to Service Differentiation: Non–Congestive Queueing", *CONWIN 2005*, July 2005

[11] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: a stateless AQM scheme for approximating fair bandwidth allocation", *INFOCOM 2000*, March 2000

[12] A. Rangarajan, and A. Acharya, "ERUF: Early Regulation of Unresponsive Best–Effort Traffic", *ICNP 1999*, November 1999

[13] I. Stoica, S. Shenker, and H. Zhang, "Core–Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks", *IEEE/ACM Transactions on Networking*, February 2003