Computer Communications 31 (2008) 3476–3484

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Vertical dimensioning: A novel DRR implementation for efficient fair queueing

Spiridon Bakiras [a], Feng Wang [b,*], Dimitris Papadias [b], Mounir Hamdi [b]

[a] Department of Mathematics and Computer Science, John Jay College of Criminal Justice, City University of New York (CUNY), United States
[b] Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Received 22 August 2007
Received in revised form 3 June 2008
Accepted 15 June 2008
Available online 24 June 2008

Keywords:
Packet scheduling
Fair queueing
Deficit round robin

ABSTRACT

Fair bandwidth allocation is an important mechanism for traffic management in the Internet. Round robin schedulers, such as Deficit Round Robin (DRR), are well-suited for implementing fair queueing in multi-Gbps routers, as they schedule packets in constant time regardless of the total number of active flows. The main drawback of these schemes, however, lies in the maintenance of per flow queues, which complicates the buffer management module and limits the sharing of the buffer space among the competing flows. In this paper, we introduce a novel packet scheduling mechanism, called Vertical Dimensioning (VD) that modifies the original DRR algorithm to operate without per flow queueing. In particular, VD is based on an array of FIFO buffers, whose size is constant and independent of the total number of active flows. Our results, both analytical and experimental, demonstrate that VD exhibits very good fairness and delay properties that are comparable to the ideal Weighted Fair Queueing (WFQ) scheduler. Furthermore, our scheduling algorithm is shown to outperform significantly existing round robin schedulers when the amount of buffering at the router is small.

1. Introduction

The fair sharing of bandwidth among competing flows inside the network is of paramount importance for efficient congestion control. The design philosophy of the Internet relies on the end-hosts to detect congestion (mainly through packet losses) and reduce their sending rates. Consequently, flows that do not respond to congestion (e.g., UDP) may end up consuming most of the available bandwidth at the expense of TCP-friendly flows, while at the same time increase the level of congestion. Fair bandwidth allocation is becoming even more important lately, due to the increasing popularity of streaming applications, such as Internet radio/TV. These applications require a stable throughput for a relatively long period of time, in order for the end-user to perceive an acceptable level of service quality.

Ideally, a router should be able to approximate a max–min fair allocation of the available bandwidth, i.e., each flow should be allocated as much bandwidth as possible, given that this allocation does not affect the throughput of any other flow. Fair queueing schedulers, such as Weighted Fair Queueing (WFQ) [1,2], are extremely effective in providing tight fairness guarantees. In fact, fair queueing is a very well-studied problem, and many variations have been proposed throughout the years that offer different levels of complexity and fairness (a detailed overview is given in Section 2).

Among all the packet schedulers reported in the literature, round robin algorithms (e.g., Deficit Round Robin (DRR) [3] and its variants) are probably the best candidates for incorporating fair queueing in multi-Gbps routers, as they schedule packets in constant O(1) time regardless of the total number of active flows. The main drawback of these schemes, however, lies in the maintenance of per flow queues that raises two important issues with respect to their performance. First, the complexity of the buffer management module increases with the number of active flows, since the longest queue needs to be identified for dropping packets in the presence of congestion. Second, and most important, the sharing of the buffer space among the competing flows becomes less effective with decreasing buffer size (a fact that is demonstrated in our simulation experiments), and thus the flow isolation property of fair queueing is not strictly enforced.

To further illustrate the importance of achieving efficient statistical multiplexing with small buffer space, we should briefly discuss the current design practice of commercial routers. The rule-of-thumb (based on the dynamics of TCP's congestion control mechanism) is that the amount of buffering at a router should be equal to the Bandwidth-Delay Product (BDP), i.e., the product of the average Round-Trip Time (RTT) times the link capacity. Consequently, today's core routers (with multi-Gbps links) contain buffers that can hold millions of packets. However, a recent study [4] suggests that the buffering capacity at the backbone routers could be reduced by up to two orders of magnitude without

* Corresponding author. Tel.: +85298352877.
E-mail addresses: sbakiras@jjay.cuny.edu (S. Bakiras), fwang@cs.ust.hk (F. Wang), dimitris@cs.ust.hk (D. Papadias), hamdi@cs.ust.hk (M. Hamdi).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.06.008

significantly reducing the link utilization. If this theory holds, it will have a positive impact on future communication networks. For instance, the end-to-end delay and delay jitter will be reduced, while the cost and complexity of backbone routers will be decreased dramatically.

To this end, we introduce a novel packet scheduling mechanism, called Vertical Dimensioning (VD) that modifies the original DRR algorithm so that it can operate without per flow queueing. In particular, we introduce a simple data structure for storing the incoming packets, based on an array of FIFO buffers. We illustrate that this structure has two very attractive properties compared to previous approaches: (i) it simplifies considerably the buffer management module at the router, and (ii) it enables efficient statistical multiplexing, even with very small buffer sizes. Our results, both analytical and experimental, indicate that VD exhibits very good fairness and delay properties that are comparable to the ideal WFQ scheduler. Furthermore, our scheduling algorithm is shown to significantly outperform existing round robin schedulers when the amount of buffering at the router is much smaller than the bandwidth-delay product.

In summary, the main contributions of our work are the following:

- We introduce a novel packet scheduling algorithm for fair bandwidth allocation that does not need to maintain per flow queues.
- We provide analytical results on the delay and fairness bounds of our algorithm, and investigate its performance (with simulation experiments) under various network conditions.
- We present initial results from a prototype implementation on a software router, and demonstrate VD's effectiveness in a real network environment.

The remainder of the paper is organized as follows. Section 2 overviews the related work in the area of fair queueing algorithms. Section 3 describes in detail the VD scheduling mechanism, and analyzes its performance and implementation complexity. Section 4 presents the results from the simulation experiments, while Section 5 provides some initial experimental results from a prototype implementation. Finally, Section 6 concludes our work.

2. Related work

Fair queueing schedulers may be generally classified into two categories, namely timestamp-based schedulers and round robin schedulers. Timestamp-based schedulers emulate as closely as possible the ideal Generalized Processor Sharing (GPS) [2] model, by computing a timestamp at each packet arrival that corresponds to the departure time of the packet under the reference GPS system. Packets are then transmitted based on their timestamp values, using a priority queue implementation. WFQ [1,2], Worst-case Fair Weighted Fair Queueing (WF^2Q) [5], and WF^2Q+ [6] fall into this category. In particular, WF^2Q achieves – what is called – "worst-case fairness", by only scheduling packets that would have started service under the reference GPS system. Although all the above algorithms exhibit excellent fairness and delay properties, the time complexity of both maintaining the GPS clock and selecting the next packet for transmission is $O(\log N)$, where N is the number of active flows [7].

The high complexity of GPS-based schedulers has led to a significant number of implementations that approximate fair queueing without maintaining exact GPS clock. Start-Time Fair Queueing (STFQ) [8], Self-Clocked Fair Queueing (SCFQ) [9], and Virtual Clock (VC) [10] are typical examples of schedulers that calculate timestamps in constant $O(1)$ time. However, since they need to maintain a sorted order of packets based on their timestamp values, the overall complexity is still $O(\log N)$ using a standard heap-based priority queue.

Leap Forward Virtual Clock (LFVC) [11] and Bin Sort Fair Queueing (BSFQ) [12] further reduce the complexity of the dequeue operation, by using an approximate sorting of the packets. Specifically, LFVC reduces the timestamp space to a set of integers, in order to make use of the Van Emde Boas priority queue that runs at $O(\log \log N)$ complexity. However, the Van Emde Boas tree is a very complex data structure, and its hardware implementation is not straightforward. BSFQ, on the other hand, achieves an $O(1)$ dequeue complexity, by grouping packets with similar deadlines into the same bin. Inside a bin, packets are transmitted in a FIFO order. This is a very efficient method for implementing fair queueing, but the number and the width of the bins must be properly set, in order to avoid empty bins (which will compromise the $O(1)$ dequeue complexity).

Round robin schedulers do not assign a deadline to each arriving packet, but rather schedule packets from individual queues in a round robin manner. As a result, most round robin schedulers are able to process packets with an $O(1)$ complexity, at the expense of weaker fairness and delay bounds. Deficit Round Robin [3] is probably the most well-known scheduler in this category. It improves on the round robin scheme proposed by Nagle [13], by taking into account the exact size of individual packets. Specifically, during each round, a flow is assigned a quantum size that is proportional to its weight. Since the size of the transmitted packet may be smaller than the quantum size, a deficit counter is maintained that indicates the amount of unused resources. Consequently, a flow may transmit (at each round) an amount of data which is equal to the deficit counter plus the quantum size. It is easy to notice that DRR has certain undesirable properties. First, it has poor delay guarantees, since each flow must wait for $N-1$ other flows before it gains access to the output link. Second, it increases the burstiness of the flows, since packets from the same flow may be transmitted back-to-back.

The above shortcomings of DRR have been addressed by many researchers, and several variations of DRR have been proposed. Smoothed Round Robin (SRR) [14], for instance, employs a Weight Spread Sequence to spread the quantum of each flow over the entire DRR round, thus reducing the output burstiness. Aliquem [15] introduces an Active List Management method that allows for the quantum size to be scaled down without compromising complexity. As a result, it exhibits better fairness and delay properties compared to the original DRR implementation. Finally, Stratified Round Robin (STRR) [16] and Fair Round Robin (FRR) [17] group flows with similar weights into classes, and use a combination of timestamp and round robin scheduling to improve the delay bound. In particular, they employ a deadline-based scheme for inter-class scheduling, and a variation of DRR for scheduling packets within a certain class. Both algorithms improve over the performance of DRR, with FRR providing better short-term fairness.

3. Vertical dimensioning

We first present in detail the VD scheduling algorithm, and then derive analytical results on its fairness and delay properties. In particular, Section 3.1 discusses the technical aspects of the algorithm, while Section 3.2 presents its performance bounds from a worst-case analysis. Finally, Section 3.3 outlines the space and time complexity of VD.

3.1. The algorithm

We consider a single link with capacity C that provides service to N backlogged flows. Each flow i has an associated weight $w_i \geqslant 1$,