An Approximation to Rate-Equalization Fairness with Logarithmic Complexity for QoS

Jorge A. Cobb
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080
Email: cobb@utdallas.edu

Suparn Gupta
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75080
Email: suparn.gupta@utdallas.edu

***Abstract*—Rate-guaranteed scheduling protocols ensure that packets from each input flow are forwarded at a rate no less than the rate reserved by the flow. WFQ is the classical example. Many of these protocols, including WFQ, provide both rate and fairness guarantees. In particular, they distribute unused capacity among among the flows in proportion to the reserved rate of each flow. In earlier work, we presented a scheduling algorithm that distributes unused capacity to flows whose reserved rate is the least. However, the per-packet complexity of this algorithm, known as rate-equalization fairness, is linear in the number of flows. Here, we present an algorithm that approximates rate-equalization fairness, but with only logarithmic complexity per packet.**

I. INTRODUCTION

Rate-guaranteed schedulers [4], [7], [9] is a family of protocols that is able to provide a lower bound on the forwarding rate of the packets of each flow, and also a bounded end-to-end delay. Iconic examples of this protocol family include Virtual Clock (VC) [14], [15] and Weighted Fair Queuing (WFQ) [11].

Most scheduling protocols provide both rate and fairness guarantees, such as WFQ and its variants [11] [1] [5] [8] [2]; they distribute the unused capacity among the flows in proportion to the reserved rate of each flow.

In earlier work [6], we presented an alternative form of fairness for rate-guaranteed schedulers: *rate-equalization*. Our protocol firsts distribute unused capacity to flows whose reserved rate is the least. This allocation continues until the flows with least reserved rate and the flows with the next-to-least reserved rate are given the same capacity. This continues, level by level, until, if enough unused capacity is available, all flows will receive the same capacity, and the scheduler will simply behave like Fair Queuing.

The disadvantage of this protocol is its computational complexity, which is in $O(n)$ per packet received, where n is the number of flows. In this paper, we present a protocol that approximates the behavior of rate-equalization, but with a smaller $O(\log(n))$ complexity.

II. RATE-EQUALIZATION FAIRNESS

We next overview the motivation for rate-equalizing fairness, which we introduced in [6]. On occasions, the bandwidth of a channel is not fully utilized. Under these conditions, it is possible for a flow to temporarily exceed its reserved rate, in an attempt to take advantage of bandwidth unused by other flows. We refer to this distribution of unallocated bandwidth as the *fairness method* of the protocol.

Some protocols, like Virtual Clock (VC) [15][14], do not address fairness. A consequence of this is that, if a flow exceeds its reserved rate, it may later be denied service by the scheduler, for a duration proportional to the time the flow exceeded its rate [5]. Other rate-guaranteed schedulers, such as Weighted Fair Queuing (WFQ) [11] and its variants [1] [5] [8] [2], distribute unused bandwidth among flows in proportion to the reserved rate of the flow. Specifically, the effective rate ψ_f that is given to flow f (i.e., the rate at which the scheduler actually forwards the packets of flow f) is

$$\psi_f(t) = \frac{C}{\left(\sum_{g \in B(t)} R_g\right)} \cdot R_f \geq R_f \tag{1}$$

where $B(t)$ is the set of backlogged flows at time t and C the capacity of the output channel.

Consider another flow g with $R_f = 2 \cdot R_g$. From (1),

$$(\psi_f - R_f) = 2 \cdot (\psi_g - R_g)$$

Hence, WFQ favors flows with a higher reserved rate.

In [6], we introduced an alternative fairness method in which the objective is to give every flow the same effective rate, provided enough unused bandwidth is available. The intuition behind it is the following. Flows whose applications are rate-adaptive could reserve the minimum rate possible to satisfy their QoS requirements, and thus minimize expense. Any additional bandwidth is given to flows that need it the most, i.e., flows with the least reserved rate.

A more detailed description is as follows. First, at all times, the effective rate of any flow f, ψ_f, is at least its reserved rate, R_f, i.e., $R_f \leq \psi_f$. Next, consider another flow g, where $R_f < R_g$. By definition, $R_g \leq \psi_g$. In our method, if enough unallocated bandwidth is available, ψ_f will increase until it becomes equal to R_g, and thus, ψ_f will become equal to ψ_g. Thus, flows with lower reserved rates will "catch up" to flows with larger reserved rates.

Assume that more unallocated bandwidth remains. In this case, the remaining unallocated bandwidth will be distributed equally between f and g, maintaining the relationship $\psi_f =$

978-1-4799-3780-6/14/$31.00 ©2014 IEEE

EQ Server

```
let $B(t)$ be the set of backlogged flows at time $t$;
for every flow $f$,
    if $f \notin B(t)$, then
        $\psi_{f,EQ}(t) = 0$
    else
        let $b_1, b_2, \ldots, b_m$ be the flows of $B(t)$
          ordered by increasing $R$;
        let $j$, $1 \leq j \leq m$, be the largest index such that
            $R_{b_j} \leq \frac{C - \sum_{k=j+1}^{m} R_{b_k}}{j} < R_{b_{j+1}}$;
        let $R_{EQ} = \frac{C - \sum_{k=j+1}^{m} R_{b_k}}{j}$;
        for each $k$, $1 \leq k \leq j$, $\psi_{b_k,EQ}(t) = R_{EQ}$;
        for each $k$, $j+1 \leq k \leq m$, $\psi_{b_k,EQ}(t) = R_{b_k}$;
```

Fig. 1. Rate equalization fluid server

ψ_g. If there exists another flow h, where $R_h > R_g$, then ψ_f and ψ_g increase equally until they reach R_h (assuming enough bandwidth remains), and hence, $\psi_f = \psi_g = \psi_h$.

To summarize, our fairness method attempts to give all flows the same effective rate. However, in doing so, the requirement of $R_f \leq \psi_f$ for all f must be preserved at all times.

We next describe our fairness method in a more formal way.

III. RATE-EQUALIZATION SERVER AND SCHEDULER

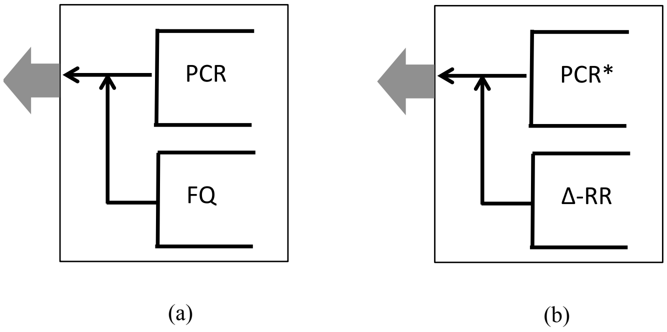
Packet scheduling algorithms that provide fairness describe their fairness method via a virtual fluid server. The packet scheduler then mimics the fluid server as much as possible. The fluid server and the packet scheduler have the same input flows, and the same output channel rate. What distinguishes them is the manner in which they forwards bits. Once the packet scheduler begins to transmit a packet, the transmission cannot be preempted. The fluid server, on the other hand, can concurrently forward an arbitrary number of bits from a group of flows; this, of course, is bounded by the capacity of the output channel.

The effective rate $\psi_f(t)$ mentioned earlier is actually the instantaneous bit rate given to f at time t by the fluid server. Thus, let $\psi_{f,EQ}(t)$ be the instantaneous bit rate given to flow f by the fluid rate-equalization server. This value is computed as shown in Figure 1.

We next focus our attention on the packet scheduler.

In general, the purpose of a fluid server is to guide the packet scheduler in the order it chooses to forward packets. Typically, [3][11][12], for every pair of packets, p_1 and p_2, if p_1 finishes service in the fluid server before p_2 finishes service, then the packet scheduler will forward p_1 before p_2. I.e., the packet scheduler tries to emulate the behavior of the fluid server as much as possible.

For most fluid servers [3][11][12], at the moment a packet arrives, the exit time that this packet will have from the fluid server is unknown. This is because the bit rate at which the packet will be served depends not only on the packets currently in the system, but also on packets that are yet to arrive. In consequence, when a packet $p_{f,i}$ arrives into a packet scheduler, the scheduler assigns to the packet a *virtual exit time* $T_{f,i}$ (see [11] for details on computing this value), such that, for any other packet $p_{g,j}$, $T_{f,i} \leq T_{g,j}$ iff the exit time of $p_{f,i}$ from the fluid server is at most the exit time of $p_{g,j}$. Packets are then forwarded in order of their virtual exit times. Thus, the packet scheduler forwards packets in the same order in which they are forwarded by the fluid server.



(a) (b)

Fig. 2. Dual-Mode Scheduling

A rate-equalizing fluid server, however, does not have this order-preserving property. That is, if two packets $p_{f,i}$ and $p_{g,j}$ are received, not only can't their exit time from the fluid server be determined, but also their relative exit times cannot be determined. I.e., which of $p_{f,i}$ or $p_{g,j}$ exits first depends on the future arrival of packets.

The lack of the order-preserving property affects the scheduling complexity. For example, $O(\log(n))$ implementations of WFQ [13] rely on this property. Thus, similar techniques cannot be applied to rate-equalization. Although we have not proven a lower bound, we speculate that a precise implementation cannot be done in $O(\log(n))$ time. We thus search for an approximation to the fluid server of rate-equalization that runs in $O(\log(n))$ time.

IV. DUAL-MODE SCHEDULING

Backlogged flows in the fluid server can be considered to be in one of two disjoint subsets: *enhanced flows*, whose effective rate is greater than their reserved rate and all members have the same effective rate, and *un-enhanced flows*, whose effective rate is simply their reserved rate. This motivates our first packet scheduler design, presented in Section IV-A. Although intuitive, this first attempt is not efficient. We then present our final scheduler design in Section IV-B

A. Flow-Migration Scheduler

Consider Figure 2(a). The service required for an un-enhanced flow f is simply a constant rate R_f. This is provided by a packetized constant rate scheduler (PCR), which is described in more detail in Figure 3. It is similar to the Virtual Clock protocol [14], except that it does not allow flows to

PCR Scheduler

```
upon receiving a packet $p_{f,i}$,
    let $S_{f,i}$ be the time when the first bit of $p_{f,i}$
    begins service at a constant rate server of rate $R_f$
    whose sole input is flow $f$;
    $F_{f,i} = S_{f,i} + L/R_f$;

if output channel is idle at time $t$,
    let $p_{f,i} \in$ Active($t$) iff $t \geq S_{f,i}$;
    if Active($t$) $\neq \emptyset$ then
        let $F_{g,j} = \min\{F_{f,i} \mid p_{f,i} \in \text{Active}(t)\}$;
        forward $p_{g,j}$ to the output channel.
```

Fig. 3. Packetized constant-rate scheduler

exceed their reserved rate. Note that this scheduler is non-work-conserving. Enhanced flows, on the other hand, have to be served in an equal manner. This is best accomplished by a fair-queuing (FQ) scheduler, also shown in Figure 2(a).

We thus have two schedulers, one for each type of flow. Priority is given to the PCR scheduler; only if the PCR scheduler is unable to provide a packet (due to its queues being empty or all packets being 'inactive'), then the packet transmitted is chosen from the FQ scheduler. Both of these schedulers can be implemented in $O(\log(n))$ time per packet arrival/departure (FQ using the method of [13]).

The above method should work, provided the membership in the enhanced and un-enhanced flow sets remains constant. However, their membership depends on unallocated bandwidth. An increase in unallocated bandwidth enhances more flows, and a decrease un-enhances some flows.

Unallocated bandwidth comes from two sources: from bandwidth that is not reserved by any flow, and from flows that have temporarily stopped creating packets (empty queues). The former is relatively stable (changes only when flows are added or removed). In this case, the appropriate movement of flows between the schedulers can be done before a new flow is accepted or removed from the system. The latter cannot be predicted, and may cause large changes in flow assignments to the two schedulers. Thus, moving flows from one scheduler to the other is not efficient, which prompts us to present below our final version of the scheduler.

B. Static-Flow-Assignment Scheduler

Our final protocol, *Approximate Packetized rate Equalization* (A-PEQ), is shown in detail in Figure 4, with an abstract view in Figure 2(b). For this implementation, we make the simplifying assumption that all packets of all flows have an equal size, L.[1] There are three major differences from the previous scheduler.

First, *all flows take part in both schedulers*. This solves the problem of having to move a large number of flows between the schedulers in a short period of time.

A-PEQ Scheduler

```
upon receiving a packet $p_{f,i}$,
    if the queue of $f$, $Q_f$, is empty, then
        let $\rho_{min} = \min\{\rho_g \mid Q_g \neq \emptyset\}$;
        $\rho_f = max(\rho_f, \rho_{min})$;
    add $p_{f,i}$ to $Q_f$;

if output channel is idle at time $t$,
    model the behavior of $PCR^*$ to dequeue a packet;
    let $p_{f,i}$ be the packet chosen by $PCR^*$;
    let $\rho_{min} = \min\{\rho_g \mid Q_g \neq \emptyset\}$;
    if $Q_f \neq \emptyset$ then
        dequeue and forward a packet from flow $f$;
        $\rho_f = \min(\rho_f + 1, \rho_{min} + \Delta)$;
    else
        let $g$ satisfy $\rho_g = \rho_{min}$;
        forward the next packet of flow $g$;
        $\rho_g = \rho_g + 1$;
```

Fig. 4. Approximate packetized rate equalization scheduler

Second, the scheduler PCR* differs from PCR as follows. PCR* assumes that every flow always has packets available (even if its queue is empty). When it chooses a packet from f for transmission, it checks the queue of f. If it is empty, then the packet transmitted is instead a packet chosen by Δ-RR. Even though f did not transmit a packet, PCR* updates its state about f as if indeed it had transmitted a packet from f.

Third, instead of FQ, we have a modified round-robin scheduling, which we denote Δ-RR. Each flow f has a round number ρ_f in Δ-RR. When Δ-RR is asked to forward a packet, it chooses it as follows.

- If Δ-RR is called because PCR* is unable to transmit a packet, then Δ-RR chooses a packet from the backlogged flow *with the least round-number*, and increases the flow's round number by one.
- If PCR* is able to transmit a packet from a flow f, then the round-number of f is increased by one, *even though* Δ-RR *did not output a packet*.

The motivation for the above choices is as follows. Consider two flows f and g, where f has a large reserved rate (always un-enhanced in the fluid server) and g a small reserved rate (always enhanced in the fluid server). All un-enhanced flows, such as f, transmit packets from PCR* at a high rate, so their round numbers in Δ-RR are higher than those of other flows. The slower flows, such as g, are served in round-robin order, and thus receive the same bandwidth.

Consider now two slow flows g and h, with g having a greater reserved rate than h. Note that through their respective packet transmissions at PCR*, the round number of g grows faster than h's. Nonetheless, both flows receive about the same behavior from Δ-RR. This is because the unused bandwidth at

---

[1] We will investigate eliminating this restriction in future work.

PCR* causes Δ-RR to serve the slowest flows, such as h, first, which allows these flows to reach the same round numbers as other flows, such as g.

One final detail remains. Assume the round number of flow f, due to its large reserved rate, grows much larger than that of other flows. Then, assume enough bandwidth becomes available to make f an enhanced flow in the fluid server. However, due its large round number, f will not receive service in Δ-RR for a long time. To avoid this, we place a bound, Δ, on the difference between the round number of any flow and the minimum round number of any backlogged flow, as indicated in Figure 4.

The bound Δ is a tunable parameter of the system. If it is too large, enhanced flows may not receive their due bandwidth, and if it is too small, bandwidth may be wasted on un-enhanced flows.

V. PERFORMANCE BOUNDS

In this section, we briefly outline some of the upper bounds on the performance of the A-PEQ scheduler. More in-depth discussions and the proofs may be found in [10]. We first note that A-PEQ is a rate-guaranteed scheduler.

Theorem 1: For every packet $p_{f,i}$ in the A-PEQ scheduler, its exit time is at most $F_{f,i,CR} + L/R_f$.

■

The reason for the term $\frac{L}{R_f}$, as opposed to the typical smaller term $\frac{L}{C}$ found in most protocols, comes from the PCR* scheduler, due to the following. PCR* chooses f, and if it finds f's queue empty, then control is passed to Δ-RR, but at this very moment a packet from f arrives. Thus, the packet has missed its scheduling opportunity in PCR*.

Next, the complexity of A-PEQ is as desired.

Theorem 2: Let n be the number of input flows to an A-PEQ scheduler. The time complexity of processing a received packet and the time complexity of selecting a packet for transmission is $O(\log(n))$.

■

For PCR*, an $O(\log(n))$ time implementation is possible using well-known techniques, such as maintaining only one finishing time, F_f, for each flow f, as opposed to maintaining one value per packet. Also, maintaining a queue of flows that will become active at some time t can be done in $O(\log(n))$ using the methods discussed in [2]. Implementing Δ-RR is obviously $O(\log(n))$, since it only needs to maintain the smallest round number among the backlogged flows.

Finally, note that, contrary to schedulers like WFQ and PEQ, A-PEQ does not simulate the behavior of the virtual fluid server. Thus, it is difficult, if not impossible, to provide an upper bound on the difference in the exit time of a packet from the EQ fluid server and the A-PEQ packet scheduler. This is why detailed simulations are presented in [13], which are omitted here due to lack of space. Nonetheless, to argue that A-PEQ does provide the desired fairness, we have the following.

Theorem 3: Assume that starting from a time t, for each flow in an A-PEQ scheduler, either its queue is always empty or always non-empty. Let $b_1, b_2, \ldots, b_m$ be the set of backlogged flows. Let j be as defined in Figure 1. Hence, flows $b_1, \ldots, b_j$ are permanently enhanced in the fluid server, while flows $b_{j+1}, \ldots, b_m$ are permanently un-enhanced. Let $P(t_1, t_2, b_k)$ be the number of bits from flow b_k transmitted by the A-PEQ scheduler during time interval $[t_1, t_2]$. Finally, let

$$\Delta > \frac{R_{max}}{R_{min}}$$

where R_{max} and R_{min} are the maximum and minimum reserved rates among the backlogged flows. Then,

- For every k, $1 \le k \le j$, as t' increases, $P(t, t', b_k)$ converges to R_{EQ}, where R_{EQ} is as defined in Figure 1.
- For every k, $j+1 \le k \le m$, as t' increases, $P(t, t', b_k)$ converges to R_{b_k}.

■

REFERENCES

[1] J. C. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 675–689, Oct. 1997.

[2] ——, "WF2Q: worst-case fair weighted fair queueing," in *IEEE INFOCOM Conference*, 1996.

[3] J. Cobb, "Universal timestamp scheduling for real-time networks," *Computer Networks*, vol. 31, pp. 2341–2360, 1999, Elsevier.

[4] J. Cobb and M. Gouda, "Flow theory," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 661–674, Oct. 1997.

[5] J. Cobb, M. Gouda, and A.-E. Nahas, "Time-shift scheduling: Fair scheduling of flows in high-speed networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, pp. 274–285, Jun. 1998.

[6] J. A. Cobb, "Rate equalization: A new approach to fairness in deterministic quality of service," *37th Annual IEEE Conference on Local Computer Networks*, vol. 0, pp. 50–57, 2011.

[7] N. Figueira and J. Pasquale, "Leave-in-time: A new service discipline for real-time communications in a packet-switching data network," in *Proc. of the ACM SIGCOMM Conference*, 1995.

[8] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *IEEE INFOCOM Conference*, 1994.

[9] P. Goyal, S. Lam, and H. Vin, "Determining end-to-end delay bounds in heterogeneous networks," in *Proc.of the NOSSDAV Workshop*, 1995.

[10] S. Gupta, "An approximation to rate-equalization fairness with logarithmic complexity for qos," Master's thesis, The University of Texas at Dallas, May, 2014.

[11] A. K. J. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, Jun. 1993.

[12] D. Stidialis and A. Varma, "Rate proportional servers: A design methodology for fair queuing algorithms," *IEEE/ACM Transactions on Networking*, Apr. 1998.

[13] P. Valente, "Exact gps simulation with logarithmic complexity, and its application to an optimally fair scheduler," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 269–280, August 2004. [Online]. Available: http://doi.acm.org/10.1145/1030194.1015497

[14] G. Xie and S. Lam, "Delay guarantee of the virtual clock server," *IEEE/ACM Transactions on Networking*, pp. 683–689, Dec. 1995.

[15] L. Zhang, "Virtual clock: A new traffic control algorithm for packet-switched networks," *ACM Transactions on Computer Systems*, vol. 9, no. 2, pp. 101–124, May 1991.