On Service Guarantees of Fair-queueing Schedulers in Real Systems

Luigi Rizzo[a], **Paolo Valente**[b]

[a] *Università di Pisa, Italy*
[b] *Università di Modena e Reggio Emilia, Italy*

Abstract

In most systems, fair-queueing packet schedulers are the algorithms of choice for providing bandwidth and delay guarantees. These guarantees are computed assuming that the scheduler is directly attached to the transmit unit with no interposed buffering, and, for timestamp-based schedulers, that the exact number of bits transmitted is known when timestamps need to be updated.

Unfortunately, both assumptions are unrealistic. In particular, real communication devices normally include FIFO queues (possibly very deep ones) between the scheduler and the transmit unit. And the presence of these queues does invalidate the proofs of the service guarantees of existing timestamp-based fair-queueing schedulers.

In this paper we address these issues with the following two contributions. First, we show how to modify timestamp-based, worst-case optimal and quasi-optimal fair-queueing schedulers so as to comply with the presence of FIFO queues, and with uncertainty on the number of bits transmitted. Second, we provide analytical bounds of the actual guarantees provided, in these real-world conditions, both by modified timestamp-based fair-queueing schedulers and by basic round-robin schedulers. These results should help designers to make informed decisions and sound tradeoffs when building systems.

Keywords: Packet scheduling, performance analysis, service guarantees

1. Introduction

Packet schedulers play a critical role in providing bandwidth and delay guarantees on non-overprovisioned transmission links. They are, e.g., one of the key components for guaranteeing the required per-client frame rate and maximum jitter in IPTV managed networks, as well as the required maximum latency in local networks for automotive and avionics applications.

An important family of packet schedulers, namely *fair-queueing* schedulers, originated for the most part as a support for the IntServ QoS architecture [10]. In these schedulers, each packet flow, identified in whatever meaningful way, is associated with a weight, and receives, in the long term, a fraction of the link bandwidth proportional to its weight. Proposed solutions range from plain round-robin [11] to accurate timestamp-based algorithms [9, 2, 1].

IntServ has basically failed as a QoS architecture in the public Internet. Nevertheless, fair-queueing schedulers, or variants of them[1], are now the algorithms of choice in most bandwidth- and delay-sensitive applications, including the previous examples. One of the reasons is that the fair-queueing service scheme easily allows both the desired minimum bandwidth to be guaranteed to each flow or aggregate, and the excess bandwidth to be evenly redistributed.

Besides, a series of very efficient yet accurate fair-queueing schedulers has been devised [13, 8, 3, 16]. All these schedulers guarantee a worst-case deviation—with respect to a perfectly fair, ideal service—comparable to that of the optimal WF^{2}Q [2] scheduler. The main practical benefits of such tight service guarantees are a very low jitter

Email addresses: rizzo@iet.unipi.it (Luigi Rizzo), paolo.valente@unimore.it (Paolo Valente)

[1] Such as *bandwidth servers* in real-time contexts.

and a smooth (non-bursty) service, as thoroughly discussed in [16].

The lowest-cost scheduler in the above series is QFQ+[16], which provides tight guarantees at the amortized cost of just Deficit Round Robin (DRR) [11]. QFQ+ has replaced its predecessor, QFQ [3], in Linux[2], and proved to be even faster than DRR, exactly in the scenarios where using an accurate scheduler matters [16].

Interestingly, the proofs in [16] are based on a slightly more complex system model than that used the in the *classical* analysis of packet schedulers. The reason why a different model has been used coincides with the motivation for this paper: on a real system, packet delays and jitters, as well as per-flow burstiness, may be much higher than predicted by classical analysis. A deeper and general investigation of this problem was out of the scope of [16], whereas it is exactly the focus of this paper[3].

Problems of classical analysis

Classical analysis is done assuming that the transmit unit is directly attached to the scheduler with no interposed buffering, and, for timestamp-based schedulers, that the exact number of bits transmitted is known on every timestamp update. Neither of these assumptions holds in practice.

First, communication links, especially high-speed ones, are equipped with FIFO queues (sometimes even large ones) to drive the device and absorb the latency and jitter in the hardware and software components that produce packets: memories, buses, interrupt service routines, etc. (in this paper we focus only on FIFO queues in output links). These FIFOs introduce an additional packet delay with any scheduler. Above all, they invalidate, by their very presence, the correctness of fair-queueing timestamp-based schedulers (Section 6.1).

Second, network interfaces do not export a real-time indication of the number of bits transmitted. Deriving this number from the time may be hard too, because, depending on the MAC protocol, the rate may vary with time.

Contributions of this paper

After illustrating the problem with a concrete example, in this paper we make the following contributions:

- We provide a simple and consistent way to modify timestamp-based, worst-case optimal and quasi-optimal fair-queueing schedulers, so as to comply with the presence of FIFOs and with uncertainty on the number of bits transmitted.

- We provide general worst-case bounds on bandwidth, packet (queueing) delay and jitter, for both the resulting family of modified schedulers and basic round-robin schedulers. These bounds take into account exactly the effects of FIFOs and uncertainty on the number of bits transmitted.

- We instantiate and compare these bounds for most schedulers in the above family as well as other popular schedulers.

As for the second contribution, we prove a good and not so obvious result: the worst-case additional packet delay caused by a FIFO, with our FIFO-compliant versions of timestamp-based schedulers, is equal at most to only the time needed to empty the FIFO, although the FIFO not only introduces the latter additional queueing delay, but also perturbs both the packet service order, as we show with a simple example in Section 2, and the timestamp computation (Section 6.1). We also prove an equivalent result in terms of service lag. In other words, we prove that FIFOs cause the minimum possible service degradation.

While partially reassuring, this result means that, in any case, sizing the output queues is critical to avoid that guarantees of sophisticated schedulers degrade to those of, e.g., DRR, or, vice versa, it means that resources should not be wasted on complex scheduling algorithms when short queues are not available. In this respect, our formulas should hopefully help designers find the right compromises between efficiency and guarantees.

Organization of this paper

The rest of this paper is structured as follows. Section 2 shows the problem through a simple example, while Section 3 briefly describes related work. Section 4 and 5 define the terms used in the paper and provide some background on timestamp-based packet schedulers. The core of the paper starts in Section 6, where we define a modification scheme that allows timestamp-based fair-queueing schedulers to comply with the presence of output

[2]QFQ is instead still available in FreeBSD.

[3]In more detail, in this paper we use the same correct model as in [16], and we report an improved version of the core proofs in [16].

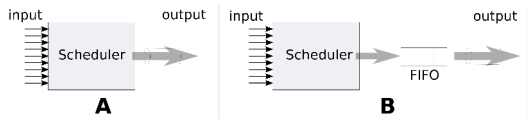

Figure 1: A: the system model used in the literature, where the scheduler drives directly an ideal link. B: a real system, made of the scheduler followed by a FIFO and the output link.

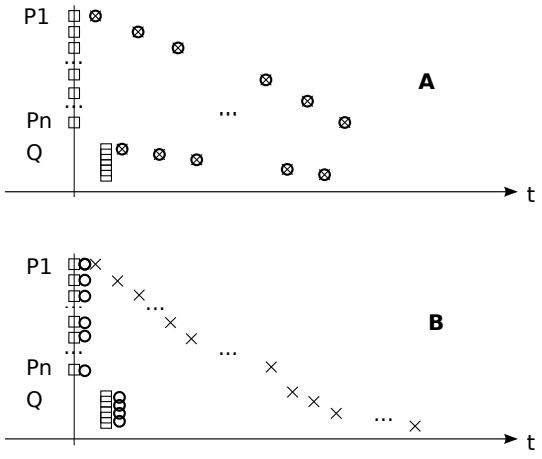

Figure 2: Timing of packets for the example in Section 2. Squares represent packet arrivals, circles are dequeues, crosses are beginnings of actual transmissions. Top: the system in Fig. 1 A shares bandwidth according to the weights of the flow. Bottom: in presence of the FIFO (Fig. 1 B), flow Q is delayed by the large backlog in the queue.

FIFOs. Section 7 then introduces the service metrics of interest. Resulting service guarantees are computed in Section 9, using the proof machinery provided in Section 8. Finally, Section 10 compares service guarantees with and without queues, and discusses practical implications of our results.

2. A simple example

We start by showing how a queue between the scheduler and the link not only introduces an obvious delay, but can also alter the service order of packets. In the system shown in Figure 1 on the left, the scheduler is directly connected to the transmit unit, which pushes bits to the communication link as soon as the scheduler makes its decision; this is the idealized model of a system normally considered in the literature. In the system on the right, the scheduler drives instead a FIFO queue which is eventually drained by the transmit unit.

Suppose that in both systems the scheduler is, e.g., WF^{2}Q+ [1], which approximates on a packet-by-packet basis an ideal, infinitely precise subdivision of the link's capacity according to flows' weights (Section 5). Suppose then that at some time t_0 a set of packets arrives simultaneously[4] for flows $P_1 \dots P_N$, all with the same weight $\phi^P = \frac{1}{2N}$. Shortly after t_0, the link becomes ready for transmission, and after another short interval a set of packets arrives for flow Q, which has weight $\phi^Q = \frac{1}{2} >> \phi^P$. For simplicity assume all packets have length L.

Figure 2 shows what happens in the system in the two configurations. In the figure, a square represents a packet arriving into the scheduler, a circle indicates a dequeue operation, and a cross the beginning of an actual transmission on the link.

The timing for the idealized case of Figure 1-A is shown in Figure 2-A. Here dequeues and beginnings of transmissions coincide. With our choice of weights, the scheduler correctly services one packet from flows P_i and one from Q, although all the flows P_i were ready before Q.

Figure 2-B shows instead the effect of a FIFO. The FIFO is instantly ready to absorb a large number of packets, so it fills up with most/all packets from flows P_i's before packets from flow Q arrive. Packets from Q therefore find a huge backlog and appear on the link only after this initial burst is complete. As a result, the transmission order and timing looks similar to the one of a DRR [11] scheduler, with or without a FIFO before the link. If the FIFO is large, then the tight service guarantees of WF^{2}Q+ have vanished, and the additional complexity in implementing a scheduler with better guarantees is completely wasted.

In addition, as shown in detail in Section 6.3, a phenomenon like that in Figure 2-B greatly perturbs timestamps in a timestamp-based scheduler. Hence it could cause further service anomalies. Fortunately, this does not happen with the schedulers considered in this paper.

[4]Such an arrival pattern is extremely realistic: on many traffic sources or routers, incoming traffic often comes in bursts (corresponding to the processing of a receive interrupt, or the generation of a large TCP segment split into packets). We assume arrivals to be exactly simultaneous for simplicity, but the problem shown in this section can be highlighted also with slightly staggered arrivals, as well as with more complex arrival patterns.

3. Related work

The analysis of timestamp-based fair-queueing packet schedulers starts with the seminal work by Parekh and Gallager [9], who show how a fluid system (GPS) can be emulated on a packet-by-packet basis using a "virtual time" concept. Subsequently, Bennet and Zhang [2] show that a simplistic emulation of a fluid system may lead to large burstiness in the output. They introduce the concept of *eligibility* and design WF^{2}Q, the first of a family of algorithms with minimal burstiness. Follow-up work include WF^{2}Q+ [1], which, using a simplified virtual time function reduces the complexity to $O(\log N)$ (the original WF^{2}Q had $O(N)$ complexity; much later, Valente [15] proved that also WF^{2}Q can be implemented in $O(\log N)$ time). Xu and Lipton [17] then prove an $\Omega(\log N)$ bound on the time complexity for any exact GPS emulation. The tradeoffs between complexity and service guarantees have been explored in a number of works [8, 13, 3, 16], presenting $O(1)$ schedulers with quasi-optimal worst-case service guarantees. Constant-time fair-queueing schedulers have also been built starting from round-robin schedulers, some of which [18] integrate concepts used in virtual-time-based schedulers.

All the papers cited so far run their analysis assuming that the number of bits transmitted is known at any time instant, and that any queueing occurs *only in the scheduler*, i.e., that once a packet leaves the scheduler, it immediately starts being transmitted on the output link. This assumption is not reflected in the reality of packet processing systems. As discussed in Section 10.3, output queues of a few hundred slots are often in use, and even if recent efforts (BQL [5], Bufferbloat [4]) try to limit software queue sizes, high-speed network interfaces may use up to 256-512 KB of internal buffering.

To our knowledge, apart from the paper introducing QFQ+ [16], which does not however investigate buffering issues in depth, there is only one work [7] that accounts for buffering after the scheduler, but it still assumes that the number of bits transmitted is known at any time, and only evaluates the time that a packet spends in the buffer, without addressing the effect of the buffer on the service properties of a scheduler.

4. Definitions and system model

In this section we report concepts and symbols used in the packet-scheduling literature, where sometimes different notations are used for the same concept. For convenience, all symbols used in this paper are listed in Table 1 (we follow the notation used in our QFQ paper [3]).

Symbol	Meaning
N	Total number of flows
$*^k$	A superscript k indicates a quantity related to flow k
$*_m$	A subscript m indicates a quantity related to a packet p_m
h, k	Flow indexes
L, L^k	Max length of any packet in the system/flow
ϕ^k	Weight of flow k
ΔW	Capacity of the FIFO in bits, plus L
l^k	Length of the head packet in flow k; $l^k = 0$ when the flow is idle
$*(t_1, t_2)$	Given a generic function $*(t)$, the notation $*(t_1, t_2) \equiv *(t_2) - *(t_1)$ indicates the difference between the values in t_2 and t_1
$W(t)$, $W^k(t)$	The "work function", i.e. number of bits transmitted (globally, or for flow k) in $[0, t]$
$V(t)$, $V^k(t)$	System/flow virtual time, see Eq. (3)
S^k, F^k, S_m, F_m	Exact virtual start and finish times of flow k or packet m, see Eq. (2)
$\hat{S}^k, \hat{F}^k$, $\hat{S}_m, \hat{F}_m$	Approximated flow/packet timestamps, see Section 5.3
$\overline{W}(t)$, $\overline{W}^k(t)$	The "work function" describing the input to the FIFO
$\overline{V}(t)$, $\overline{V}^k(t)$	Inflated system/flow virtual time corresponding to $\overline{W}(t)$
$\overline{S}^k, \overline{F}^k$	Inflated virtual start and finish times, computed through (8)
$\tilde{S}^k, \tilde{F}^k$	Counterparts of $\hat{S}^k$ and $\hat{F}^k$, obtained from $\overline{S}^k, \overline{F}^k$ instead of from S^k, F^k
$B(t)$	The set of backlogged flows at time t
$Q^k(t)$	Backlog of flow k at time t

Table 1: Definitions of the symbols used in the paper.

For ease of exposition, we often use the notation

$$f(t_1, t_2) \equiv f(t_2) - f(t_1)$$

where $f(t)$ is a function of the time.
We assume that any discontinuous function of the time is left-continuous, i.e., if t_0 is a discontinuity point for a function $f(t)$, then $f(t_0) = \lim_{\epsilon \to 0} f(t_0 + |\epsilon|)$, and $f(t_0^-) = \lim_{\epsilon \to 0} f(t_0 - |\epsilon|)$.

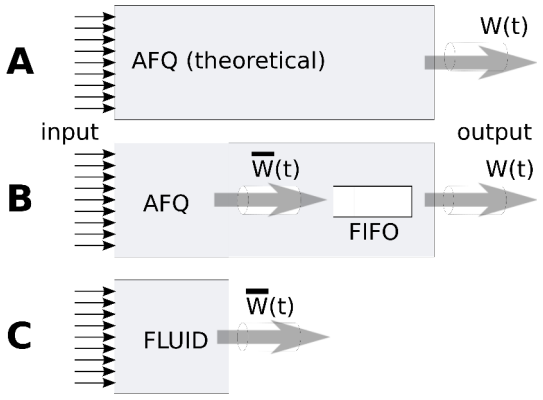

Figure 3: A: the system model used in the literature, ignoring the presence of the FIFO and assuming $W(t)$ is known exactly within the scheduler. The scheduler box is labelled as AFQ, which is the reference scheduler in this paper (defined in Section 5.3). B: a real system, made of a scheduler (AFQ) feeding a dequeue unit with work function $\overline{W}(t)$, followed by a FIFO and the output link. C: the corresponding fluid system for the first part of B), serving multiple packets at a time, with the same work function $\overline{W}(t)$.

4.1. System model

As a starting point, we consider a system as in Figure 3 A, in which N packet flows (defined in whatever meaningful way) share a common, non-preemptive transmission link, serving one packet at a time. The link has a time-varying rate, with $W(t)$ being its *work function*, i.e., the total number of bits transmitted in $[0, t]$. A packet scheduler, the AFQ block in the figure (we define the AFQ scheduler in Section 5.3), sits between the flows and the link: arriving packets are immediately enqueued, and the next packet to serve is chosen and dequeued by the scheduler when the link is ready.

Each flow k is assigned a fixed weight $\phi^k > 0$. Without losing generality, we assume that $\sum_{k=1}^{N} \phi^k \leq 1$. A flow is defined *backlogged/idle* if it owns/does not own packets not yet completely transmitted. Each flow uses a FIFO queue to hold its backlog. We call $B(t)$ the set of flows backlogged at time t.

We call *head packet* of a flow the packet at the head of the queue, and l^k its length; $l^k = 0$ when a flow is idle. We say that a flow is *receiving service* if one of its packets is being transmitted. Both the amount of service $W^k(t_1, t_2)$ received by a flow and the total amount of service $W(t_1, t_2)$ delivered by the system in the time interval $[t_1, t_2]$ are measured in number of bits transmitted during the interval.

We say that a system is *work conserving* if it uses the link at full capacity whenever there are packets queued. Finally, two systems are *corresponding* [2, Definition 1] if they: have the same work function $W(t)$, serve the same set of flows, are subject to the same arrival pattern.

5. WF^{2}Q+ and its approximated variants

In this section we outline the schedulers we focus on in this paper, namely WF^{2}Q+ and its approximated variants. In particular, we consider the more general case of a variable-rate link (see [1, 14] for a complete description). WF^{2}Q+ is an online work-conserving *packet scheduler* that approximates, on a non-preemptive, packet-by-packet basis, the service provided by a corresponding work-conserving *ideal fluid system*. The latter may serve multiple packets in parallel, and delivers the following, almost perfect bandwidth distribution over any time interval during which a flow is continuously backlogged:

$$W^k(t_1, t_2) \geq \phi^k W(t_1, t_2) - (1 - \phi^k)L \qquad (1)$$

To define the WF^{2}Q+ algorithm, we need to introduce the concept of *eligibility*: a packet is defined as *eligible* at a given time instant if it has already started in the fluid system by that time. Accordingly, we define a flow as eligible if its head packet is eligible.

WF^{2}Q+ works as follows: when requested to provide the next packet to serve, it chooses and dequeues, among the eligible packets, the next one that would be completed in the fluid system if no other packet arrived or became eligible; ties are arbitrarily broken. WF^{2}Q+ therefore succeeds in finishing packets in the same order as the ideal fluid system, except when the next packet to serve arrives or becomes eligible after one or more out-of-order packets have already started.

5.1. Virtual Times

The WF^{2}Q+ policy is efficiently implemented by considering, for each flow, a special *flow virtual time* function $V^k(t)$ that grows as the *normalized* amount of service received by the flow, i.e., by the actual service received, divided by the flow's weight. Besides, when the flow turns from idle to backlogged, $V^k(t)$ is set to the maximum between its current value and the value of a further function, the *system virtual time* $V(t)$, defined below.

In addition to $V^k(t)$, each flow is conceptually[5] associated with a virtual time $V^k_{fluid}(t)$ also in the fluid system. $V^k(t)$ and $V^k_{fluid}(t)$ are computed with the same rules, but their values differ as the instantaneous distribution of work is different in the packet and in the corresponding fluid system.

For every packet of flow k, we define the virtual *start* and *finish time* of the packet as the value of $V^k(t)$ when the packet starts and finishes to be served. We then define the *virtual start* and *finish times* of flow k, $S^k(t)$ and $F^k(t)$, as the virtual start and finish times of its head packet at time t. These timestamps need to be updated only when the flow becomes backlogged, or when its head packet is dequeued:

$$S^k(t_p) \leftarrow \begin{cases} \max(V(t_p),\, F^k(t_p^-)) & \text{on newly backlogged flow;} \\ F^k(t_p^-) & \text{on packet dequeue;} \end{cases}$$
$$F^k(t_p) \leftarrow S^k(t_p) + l^k/\phi^k \tag{2}$$

where t_p is the time when a packet enqueue/dequeue occurs, and l^k is the packet size. Finally, the system virtual time function $V(t)$ is defined as follows (assuming $\sum_{k=1}^{N} \phi^k \leq 1$):

$$V(t_2) \equiv \max\left\{ V(t_1) + W(t_1, t_2),\, \min_{k \in B(t_2)} S^k(t_2) \right\} \tag{3}$$

Note that the instantaneous link rate needs not be known to update $V(t)$: just $W(t_1, t_2)$ (the total number of bits transmitted in $[t_1, t_2]$) suffices. At system start up, $V(0) = 0$, $S^k(0) \leftarrow 0$ and $F^k(0) \leftarrow 0$.

5.2. Implementation of WF^{2}Q+

The fluid system guarantees that $V^k_{fluid}(t) \geq V(t)$ always holds for every backlogged flow k [1, 14]. Hence, in terms of virtual times, flow k is *eligible* at time t if $V(t) \geq S^k(t)$. In addition, the fluid system serves flows so as to complete packets in virtual-finish-time order. WF^{2}Q+ can then be implemented as follows, using only $V(t)$ and the virtual start and finish times of the flows: each time the next packet to transmit is requested, the scheduler dequeues and returns the head packet of the eligible flow with the smallest virtual finish time. The second argument of the max operator in Eq. (3) guarantees that the system is work-conserving.

[5] This parameter is not needed in the implementation, but we use it to prove Lemma 1.

5.3. Approximated variants of WF^{2}Q+ (AFQ)

The exact WF^{2}Q+ algorithm, as described above, has $\Omega(\log N)$ complexity in the number of flows [17]. In order to implement the same policy in $O(1)$ time, several schedulers [8, 13, 3] label flows with approximated virtual start and finish times $\hat{S}^k(t)$ and $\hat{F}^k(t)$, in addition to the exact timestamps defined in Eq. (2). Lowest-cost examples are QFQ [3], S-KPS [8] and the scheduler proposed in [13], which we call GFQ hereafter.

In these schedulers, approximated timestamps are used to choose the next packet to transmit (Section 5.2), and to compute the system virtual time, i.e., $\hat{S}^k(t_2)$ is used instead of $S^k(t_2)$ in Eq. (3). Using approximated values helps reduce the complexity of sorting stages, making them constant-time operations. Exact timestamps are instead still used to charge flows for the service received (Eq. (2)).

The way approximations are computed varies with the scheduler, but in all cases we can write

$$\begin{aligned} S^k(t) - \Delta S^k \leq \hat{S}^k(t) \leq S^k(t) \leq \\ F^k(t) \leq \hat{F}^k(t) \leq F^k(t) + \Delta F^k \end{aligned} \tag{4}$$

where ΔS^k and ΔF^k are non-negative quantities ($\Delta S^k = \Delta F^k = 0$ in WF^{2}Q+). For brevity, hereafter we use the generic name AFQ (Approximated Fair Queueing), to refer to any of these variants and to WF^{2}Q+ itself.

6. Moving to a real system

The algorithms described in Section 5 work correctly in the *classical* system model in Figure 3 A. Unfortunately, as we show as a first step in this section, that model is inadequate for most real systems. We address this issue, in the rest of this section, by showing both a way to properly extend the model, and a way to modify existing AFQ schedulers so as to preserve the correctness of their operations in the new model.

6.1. Mismatch between a real system and the classical model

Packet schedulers typically operate at Layer 3, i.e., right above network-interface drivers. In this respect, with typical network interface controllers (*NICs*), the there are at least two queues between the scheduler and the transmit unit (see, e.g., [6]):

- a software FIFO, typically called *transmit ring*, used as a mailbox between the OS and the NIC;

- a hardware FIFO, internal to the NIC and often called *packet buffer*.

These queues are essential to make sure that the link does not remain idle after the transmission of a packet has finished. They are manipulated as follows. In parallel with packet transmissions, the OS dequeues packets from the scheduler and inserts them into the transmit ring, whereas the NIC fetches packets from the transmit ring and inserts them into its hardware queue. Only after being inserted into the hardware queue, a packet is finally ready to be transmitted over the physical medium.

The transmit ring allows the NIC to always have packets ready to fetch when needed, independently of possible slowness in the OS, whereas the insertion of packets into the hardware FIFO before transmission makes sure that after the transmission of a packet has started, it is not aborted because the NIC cannot read data from memory due to bus contention. In this respect, the size of the hardware FIFO is at least L, to have room for at least one packet. The size of the hardware FIFO is actually likely to be larger, to have also the next packet(s) enqueued, and therefore immediately ready for transmission, before the current one is finished.

The above queues are often filled and drained in bursts, as traffic arrives or low/high water marks are reached. In the end, packet dequeues from the scheduler can occur at any time. In addition: (a) NICs operate on a packet-by-packet basis, and do not export a real-time indication of the number of bits transmitted, (b) the data rate is not constant due to framing, link contention and link-level flow control. Even the notification of transmission completions, available through memory-mapped registers or interrupts, can be delayed by several microseconds, corresponding to tens/hundreds of packets on high speed links. In the end, $W(t)$ is not known at all times, whereas packet enqueues and, as already said, dequeues, may occur at any time. Hence both $V(t)$ and flow timestamps may need to be updated at any time.

Because of the above two facts, **the following two assumptions are almost never true in a real system**: (i) the link may request a new packet to transmit only once the previous one has been fully transmitted, (ii) the exact value of $W(t)$ is always known when it is needed to update $V(t)$ in Eq. (3).

The operating problems caused by the failure of the second assumption are evident. Unfortunately, the failure of the first assumption causes critical problems as well. For an AFQ scheduler to produce a correct schedule, the system virtual time computed through Eq. (3) *must* be a lower bound to the normalized service provided to any backlogged flow by the fluid system [1]. And the first assumption is an implicit, necessary condition for this property to hold. In fact, intuitively, dequeueing packets *too early* with respect to the total work done by the system may cause the virtual start time of some backlogged flow, computed by Eq. (2), to become too high with respect to the normalized service actually received by the flow, and hence $V(t)$ may *jump* to incorrectly high values by Eq. (3).

6.2. Model extension

A realistic model of a communication device is the one in Figure 3 B, where the scheduler is drained by a dequeue unit that takes care of inserting packets in a FIFO queue. In particular, for simplicity, we use a single FIFO to represent all the buffering between the scheduler and the link. Finally, we assume that the time needed to insert a packet in the FIFO is negligible with respect to the transmission time of the same packet on the link.

We denote as $\overline{W}(t)$ and $\overline{W}^k(t)$ the sum of the sizes of, respectively, all the packets and only the packets of flow k dequeued from AFQ during $[0, t]$. In other words, $\overline{W}(t)$ is the amount of work *delivered to the FIFO* up to time t. The function $\overline{W}(t)$ has a stepwise shape (Figure 4), and lies in a band of height ΔW on top of $W(t)$, i.e.,

$$W(t) \leq \overline{W}(t) \leq W(t) + \Delta W, \tag{5}$$

where ΔW equals the maximum capacity of the FIFO. The latter is at least L (as this is the minimum size of the hardware FIFO, as highlighted in Section 6.1), i.e.,

$$\Delta W = \max_t \overline{W}(t) - W(t) \geq L. \tag{6}$$

6.3. Modification of the original AFQ scheduler

A way to modify AFQ so as to work correctly also in the new model is to use the approximate value $\overline{W}(t)$ as the work function, instead of the exact value of $W(t)$. Since, by Eq. (5), $\overline{W}(t) \geq W(t)$ holds, we have that Eq. (3) now yields an *inflated*

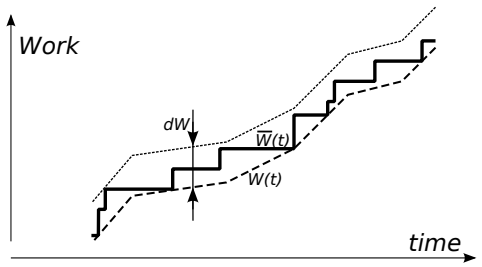

Figure 4: $\overline{W}(t)$, the number of bits extracted from the scheduler, is within a band of height ΔW above $W(t)$, the number of bits transmitted by the link.

system virtual time, which we denote as $\overline{V}(t)$. In particular, for AFQ, Eq. (3) now becomes

$$\overline{V}(t_2) \equiv \max\left\{\overline{V}(t_1) + \overline{W}(t_1, t_2), \min_{k \in B(t_2)} \tilde{S}^k(t_2)\right\} \tag{7}$$

for which we explain in a moment what $\tilde{S}^k(t_2)$ represents. We also get *inflated* flow virtual times $\overline{V}^k(t)$ (Section 5.1), and, using $\overline{V}(t)$ instead of $V(t)$ in Eq. (2), we get the following *inflated* virtual start and finish timestamps:

$$\begin{aligned} \overline{S}^k(t_p) &\leftarrow \begin{cases} \max(\overline{V}(t_p), \overline{F}^k(t_p^-)) & \text{on newly backlogged flow;} \\ \overline{F}^k(t_p^-) & \text{on packet dequeue;} \end{cases} \\ \overline{F}^k(t_p) &\leftarrow \overline{S}^k(t_p) + l^k/\phi^k \end{aligned} \tag{8}$$

As we explained in Section 5.3, AFQ is characterized by the use of the approximated timestamps $\hat{S}^k(t)$ and $\hat{F}^k(t)$. We denote as $\tilde{S}^k(t)$ and $\tilde{F}^k(t)$ the counterparts of these timestamps for the modified AFQ. More precisely, $\tilde{S}^k(t)$ and $\tilde{F}^k(t)$ are computed, for each instance of AFQ (such as QFQ, S-KPS or GFQ), in the same way as $\hat{S}^k(t)$ and $\hat{F}^k(t)$. But, differently from $\hat{S}^k(t)$ and $\hat{F}^k(t)$, $\tilde{S}^k(t)$ and $\tilde{F}^k(t)$ are computed as a function of $\overline{S}^k(t)$ and $\overline{F}^k(t)$ instead of $S^k(t)$ and $F^k(t)$. Therefore, from Eq. (4), we get

$$\begin{aligned} \overline{S}^k(t) - \Delta S^k \leq \tilde{S}^k(t) \leq \overline{S}^k(t) \leq \\ \overline{F}^k(t) \leq \tilde{F}^k(t) \leq \overline{F}^k(t) + \Delta F^k \end{aligned} \tag{9}$$

Finally, according to how the system virtual time is computed in AFQ (Section 5.3), $\tilde{S}^k(t_2)$ is used instead of $\overline{S}^k(t_2)$ in (7).

On each packet dequeue, the work function $\overline{W}(t)$ used in Eq. (7) immediately grows by the size of the packet, i.e., by the same quantity by which $W(t)$ grows while the same packet is transmitted. Therefore, for the resulting new timestamp-computation logic, it is like if every packet is completely and instantaneously transmitted on each dequeue.

We can model this fact with the presence of an infinite-speed link before the FIFO in Figure 3 B. This allows us to define a *fictitious* subsystem of that in Figure 3 B, made of the cascade of the scheduler and of a fictitious, infinite-speed link. Such a link happens to have fully transmitted the previous packet before a new one is requested. Since, in addition, the value of $\overline{W}(t)$ is of course known at all times, both the assumptions in Section 6.1 do hold for this fictitious (sub)system.

As a consequence, by the classical theory, the modified scheduler happens to correctly approximate the service provided by the fluid system corresponding to this fictitious system, namely the fluid system in Figure 3 C. The modified scheduler thus provides, for the fictitious system, the same *classical* service guarantees as the original scheduler for the simplified system in Figure 3 A. We turn this important fact into some useful relations in Section 8, and then, in Section 9, we use these relations to compute the actual service guarantees of the modified scheduler in a real system with an interposed FIFO. For brevity, hereafter we call just AFQ its modified version.

We conclude this section by observing that $\overline{W}(t)$ is the closest approximation of $W(t)$ for which there exists at least a fictitious system (the above-described one) for which an AFQ scheduler using an approximate work function operates correctly. In fact, if the first assumption in Section 6.1 does not hold in a real system, i.e., if new packets can be dequeued before the previous ones have been fully transmitted, then it is easy to see that the same assumption does not hold even in any fictitious system characterized by the same packet-dequeue pattern (from the scheduler) as the real system, and by a work function that may have lower values than $\overline{W}(t)$ at some times t.

7. Service metrics

We introduce now the two service metrics by which we show and compare the ideal (according to classical analysis) and the real service guarantees provided by packet schedulers. The first metric concerns packet delay and jitter, whereas the second one concerns short- and long-term bandwidth

distribution. For both metrics, we first report their formal definition, and then discuss their meaning.

There is a third useful metric (RFI [12]), which measures how evenly excess bandwidth is redistributed. To keep the description of the results more concise, in the body of the paper we focus only on the first two metrics. We compute however bounds also for the last metric in the Appendix.

7.1. T-WFI

For a link with a constant rate R, i.e., such that $W(t_1, t_2) = R(t_2 - t_1)$ if there is backlogged traffic during $[t_1, t_2]$, the Time Worst-case Fair Index [2] T-WFI^k for flow k is defined as

$$\text{T-WFI}^k \equiv \max\left(t_c - t_a - \frac{Q^k(t_a)}{\phi^k R}\right) \qquad (10)$$

where t_a and t_c are, respectively, the arrival and completion time of a packet, and $Q^k(t_a)$ is the backlog of flow k just after the arrival of the packet. For a variable-rate link, R may be interpreted as the average rate during $[t_a, t_c]$.

T-WFI^k is then equal to the maximum possible delay with respect to the ideal, worst-case completion times of the packets of flow k, according to the weight of the flow. As such, it is not only a fairness index, but also a direct measure of worst-case packet delay and delay variation (jitter). In this respect, as shown experimentally in [16], observed maximum and average packet delays/jitters, with respect to a perfectly fair and smooth service, are in the order of exactly the T-WFI. As a consequence, only schedulers with a tight T-WFI (e.g., in the order of the packet transmission time at the rate reserved to the flow [16]) can guarantee a smooth service. On the opposite end, depending on the scenario, schedulers with a loose T-WFI may suffer from such a high packet delay and jitter to make time-sensitive applications unfeasible [16].

7.2. B-WFI

The Bit Worst-case Fair Index (B-WFI), introduced in [1], is in a sense the counterpart of the T-WFI in terms of amount of service. Unlike the T-WFI, it can be computed even if the link rate is unknown (and hence also if it is variable). The B-WFI^k for a flow k is defined as:[6]

$$\text{B-WFI}^k \equiv \max_{[t_1,t_2]} \left\{\phi^k W(t_1, t_2) - W^k(t_1, t_2)\right\} \quad (11)$$

[6]This definition is slightly more general than the original one in [1], where t_2 was constrained to the completion time of a packet.

where $[t_1, t_2]$ is any time interval during which the flow is continuously backlogged, $\phi^k W(t_1, t_2)$ is the minimum amount of service the flow should have received according to its share of the link bandwidth, and $W^k(t_1, t_2)$ is the actual amount of service provided by the scheduler to the flow.

Since B-WFI^k takes into account any possible time interval during which flow k is backlogged, it reveals first whether the flow receives its share of the bandwidth in the long term (in which case B-WFI^k is constant). Secondly, if the latter condition is satisfied, then B-WFI^k measures the extent at which the flow still suffers from service fluctuations, and hence burstiness, in the short term. In practice, similarly to the T-WFI, only schedulers with a tight B-WFI (e.g., in the order of a few packets) are likely to guarantee a low-enough burstiness to comply with the requirements of most time-sensitive applications [16].

8. Proof machinery

In Section 9 and in Appendix B, we compute the T-WFI, B-WFI and RFI for AFQ, in systems modeled as in Figure 3 B. We compute these service metrics as a function of lower bounds to the number of bits $\overline{W}^k(t_1, t_2)$ that AFQ guarantees to be dequeued for any flow while backlogged. This approach, being $\overline{W}^k(t_1, t_2)$ equal to the service guaranteed to the flow by the fictitious system defined in Section 6.3, allows us to split the problem of computing the above metrics, which take into account also the effects of the FIFO, into two simpler subproblems: computing the needed bounds to the service $\overline{W}^k(t_1, t_2)$ guaranteed by the fictitious system, which has no FIFO, and computing the needed relations between the metrics and $\overline{W}^k(t_1, t_2)$. In particular, for the fictitious system, classical analysis can still be used (Section 6.3).

In this section we provide the machinery for solving both sub-problems. In more detail, in Section 8.1 we tackle the second sub-problem for the B-WFI, by providing a special relation concerning the quantity $W^k(t_1, t_2)$. Simpler relations are needed for the other metrics, and are reported directly in the proofs of the metrics. In sections 8.2 and 8.3, we tackle instead the first sub-problem, by reporting a sequence of lemmas that culminates in the bounds to $\overline{W}^k(t_1, t_2)$. Finally, in Section 8.4 we report a general bound for DRR, which we use in Section 10 to compute the T-WFI and the B-WFI

also for DRR.

Notation: For the reader's convenience, on top of various equality or inequality signs we write the reason why the relation holds (typically a reference to another relation).

8.1. Relation needed for the B-WFI

The relation we use to compute the B-WFI can be stated informally as follows: the service $W^k(t_1,t_2)$ received by flow k during $[t_1,t_2]$ is equal to the number of bits dequeued for flow k, $\overline{W}^k(t_1,t_2)$, minus the bits of flow k still in the FIFO or in service at time t_2. To express this relation as a formula, we start by denoting as $\hat{t}$ the largest time instant such that $\hat{t} \leq t_2$ and all packets of flow k that have been dequeued during $[\hat{t},t_2]$ are still in the FIFO or in service at time t_2. By definition, the sum of the sizes of all these packets is $\overline{W}^k(\hat{t},t_2)$, and we have

$$0 \leq \overline{W}^k(t_2) - W^k(t_2) \leq \overline{W}^k(\hat{t},t_2) \qquad (12)$$

Using these two inequalities, we can then write:

$$\begin{aligned} W^k(t_1,t_2) = W^k(t_2) - W^k(t_1) \geq \\ \max\{0, W^k(t_2) - \overline{W}^k(t_1)\} \geq \\ \max\{0, \overline{W}^k(t_2) - \overline{W}^k(\hat{t},t_2) - \overline{W}^k(t_1)\} = \\ \max\{0, \overline{W}^k(t_2) - \overline{W}^k(t_2) + \overline{W}^k(\hat{t}) - \overline{W}^k(t_1)\} = \\ \max\{0, \overline{W}^k(\hat{t}) - \overline{W}^k(t_1)\} = \overline{W}^k(t_1, \max\{t_1,\hat{t}\}) \end{aligned} \qquad (13)$$

8.2. Globally-Bounded Timestamps

The *Globally-Bounded Timestamp* property (GBT) [13, Definition 3] states that flow timestamps cannot deviate too much from the system's virtual time. Here we compute a variant for approximate timestamps of this property that comes in handy to lower-bound $\overline{W}^k(t_1,t_2)$. The proofs are essentially the same as in the literature [13], and are reported in Appendix A.

Lemma 1 (Lower bound to $\tilde{F}^k(t)$). *For all times t at which flow k is backlogged,*

$$\overline{V}(t) - \tilde{F}^k(t^-) \leq L \qquad (14)$$

Lemma 2 (Upper bound to $\overline{S}^k(t)$). *At all times t*

$$\overline{S}^k(t) \leq \overline{V}(t) + \Delta S^k + \frac{L^k}{\phi^k} - L^k \qquad (15)$$

Note: the last two bounds apply to timestamps with a different degree of approximation (Section 6.3). This differentiation comes in handy in the proof of subsequent Lemma 3.

8.3. Lower bounds to $\overline{W}^k(t_1,t_2)$

This subsection contains two lower bounds to $\overline{W}^k(t_1,t_2)$, one expressed in terms of the virtual time $\overline{V}(t)$ and the other in terms of the work function $W(t)$[7]. We use the former in the derivation of the RFI, and the latter in the derivation of the T-WFI and the B-WFI.

Lemma 3. *We have, in general,*

$$\begin{aligned} \overline{W}^k(t_1,t_2) \geq \\ \phi^k \overline{V}(t_1,t_2) \\ -\phi^k \left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + L - L^k \right) \end{aligned} \qquad (16)$$

whereas, if t_2 coincides with the completion time of a packet of flow k, then the term $2\frac{L^k}{\phi^k}$ can be replaced with $\frac{L^k}{\phi^k}$ in the above bound.

Proof. Recalling the meaning of the virtual time $\overline{V}^k(t)$ of flow k, we can write the following equalities, where the last equality follows from summing and subtracting $\overline{V}(t_2) - \overline{V}(t_1)$ to $V^k(t_2) - \overline{V}^k(t_1)$:

$$\begin{aligned} \overline{W}^k(t_1,t_2) = \phi^k \overline{V}^k(t_1,t_2) = \\ \phi^k \left[\overline{V}^k(t_2) - \overline{V}^k(t_1) \right] = \\ \phi^k \left[(\overline{V}(t_2) - \overline{V}(t_1) \right] + \\ \phi^k \left[\overline{V}^k(t_2) - \overline{V}(t_2) - (\overline{V}^k(t_1) - \overline{V}(t_1)) \right] \end{aligned} \qquad (17)$$

We can therefore prove the thesis by computing lower bounds to the two terms $\overline{V}^k(t_2) - \overline{V}(t_2)$ and $-(\overline{V}^k(t_1) - \overline{V}(t_1))$. Remembering that by definition

[7] Remembering that by definition $\overline{W}^k(t_1,t_2) \geq 0$, we could derive tighter bounds by writing $\overline{W}^k(t_1,t_2) \geq \max\{0,...\}$. However this would make the result even less readable, and it is hardly useful given that the equation is later used in a context where we take the maximum over any flow and/or time intervals.

$\overline{V}^k(t) = \overline{S}^k(t)$, for the first term we have

$$\begin{aligned} \overline{V}^k(t_2) - \overline{V}(t_2) = \overline{S}^k(t_2) - \overline{V}(t_2) &\overset{(8) \text{ and } l^k \le L^k}{\geq} \\ \overline{F}^k(t_2) - \tfrac{L^k}{\phi^k} - \overline{V}(t_2) &\overset{(9)}{\geq} \\ \tilde{F}^k(t_2) - \tfrac{L^k}{\phi^k} - \Delta F^k - \overline{V}(t_2) &\overset{(14)}{\geq} \\ -\tfrac{L^k}{\phi^k} - \Delta F^k - L \end{aligned} \tag{18}$$

But, if, in particular, t_2 coincides with the completion time of a packet of flow k, then we have, more precisely that $\overline{V}^k(t) = \overline{S}^k(t) = \overline{F}^k(t^-)$ from (8). Repeating the same derivations as above, but starting from $\overline{V}^k(t) = \overline{F}^k(t^-)$, we get, for this special case,

$$\overline{V}^k(t_2) - \overline{V}(t_2) \geq -\Delta F^k - L \tag{19}$$

As for the second term, we have

$$\begin{aligned} -\left[\overline{V}^k(t_1) - \overline{V}(t_1)\right] = \\ -\left[\overline{S}^k(t_1) - \overline{V}(t_1)\right] \overset{(15)}{\geq} \\ -\left[\overline{V}(t_1) + \Delta S^k + \frac{L^k}{\phi^k} - L^k - \overline{V}(t_1)\right] = \\ -\left[\Delta S^k + \frac{L^k}{\phi^k} - L^k\right] \end{aligned} \tag{20}$$

Replacing the bounds (18), or (19), and (20) in (17), and rearranging terms, we get the thesis.

Lemma 4. *We have, in general,*

$$\begin{aligned} \overline{W}^k(t_1,t_2) \geq \phi^k \left(\overline{W}(t_2) - W(t_1)\right) + \\ -\phi^k \left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + L - L^k + \Delta W\right) \end{aligned} \tag{21}$$

whereas, if t_2 coincides with the completion time of a packet of flow k, then the term $2\frac{L^k}{\phi^k}$ can be replaced with $\frac{L^k}{\phi^k}$ in the above bound.

Proof. We prove the thesis by upper-bounding the term $\overline{V}(t_1,t_2)$ in (16) as follows:

$$\begin{aligned} \overline{V}(t_2) - \overline{V}(t_1) \overset{(7)}{\geq} \overline{W}(t_2) - \overline{W}(t_1) &\overset{(6)}{\geq} \\ \overline{W}(t_2) - W(t_1) - \Delta W \end{aligned} \tag{22}$$

8.4. Service guarantees of DRR

We define as *transmission opportunity* for flow k every maximal sub-interval of $[t_1,t_2]$ during which the flow is continuously at the head of the DRR queue. Let h be the number of transmission opportunities for flow k during $[t_1,t_2]$. If t_1 is lower or equal to the beginning of the first transmission opportunity, and if we denote as ϕ_{min} the minimum possible weight among flows, from [11, Lemma 2], we can derive the following loose upper bound (independent of ΔW):

$$\overline{W}(t_1,t_2) < \frac{h}{\phi_{min}} L + (N-1)L \tag{23}$$

We use this loose bound instead of a tighter one to avoid longer formulas. As for the service received by flow k, we have, again from [11, Lemma 2] (and again regardless of ΔW),

$$\overline{W}^k(t_1,t_2) \geq (h-1)\frac{\phi^k}{\phi_{min}} L - L \tag{24}$$

where the factor $h-1$ in the first term is not equal to h because t_2 may be, in the worst case, equal to the beginning of the h-th transmission opportunity, and hence flow k may have not yet used that opportunity at all by time t_2.

We want now to upper-bound $\overline{W}(t_1,t_2)$ as a function of $\overline{W}^k(t_1,t_2)$. To this purpose, solving (24) for h, and replacing the result in (23), we get

$$\overline{W}(t_1,t_2) \leq \frac{\overline{W}^k(t_1,t_2)}{\phi^k} + \left(\frac{1}{\phi_{min}} + \frac{1}{\phi^k} + N - 1\right) L \tag{25}$$

9. Service properties of AFQ

In this section we compute the T-WFI and the B-WFI for AFQ, using the intermediate bounds proved in the previous section. Formulas are little intuitive at a first glance, but it is easy to show what each term accounts for, as we do in Section 9.1, after reporting and proving formulas themselves.

Theorem 1 (T-WFI). *For a flow k, AFQ guarantees*

$$T\text{-}WFI^k = \frac{L^k}{\phi^k R} + \frac{\Delta S^k + \Delta F^k + L - L^k + \Delta W}{R} \tag{26}$$

PROOF. Given a packet p arriving at time t_a, we prove the thesis in two steps: first we compute an upper bound to the time that elapses from t_a to when p is dequeued from AFQ, say time $\overline{t}_c$, then we add to this upper bound the maximum time that may elapse from time $\overline{t}_c$ to the time instant t_c at which p is finally transmitted.

As for the first step, by definition of $\overline{W}^k(t)$ and $\overline{t}_c$, we have that $\overline{W}^k(t_a, \overline{t}_c) = Q^k(t_a)$. Using this equality and (21), and recalling that the link works at constant speed R, we can write

$$\begin{aligned}
\overline{t}_c - t_a \leq \frac{W(t_a, \overline{t}_c)}{R} = \\
\frac{W(t_a, \overline{t}_c) + \overline{W}(\overline{t}_c) - \overline{W}(\overline{t}_c)}{R} = \\
\frac{\overline{W}(\overline{t}_c) - W(t_a) + W(\overline{t}_c) - \overline{W}(\overline{t}_c)}{R} \overset{\text{Lemma 4}}{\leq} \\
\frac{W(\overline{t}_c) - \overline{W}(\overline{t}_c)}{R} + \frac{\overline{W}^k(t_a, \overline{t}_c)}{\phi^k R} + \\
\frac{\frac{L^k}{\phi^k} - L^k + \Delta S^k + \Delta F^k + L + \Delta W}{R} = \\
\frac{W(\overline{t}_c) - \overline{W}(\overline{t}_c)}{R} + \frac{Q^k(t_a)}{\phi^k R} + \\
+ \frac{\frac{L^k}{\phi^k} - L^k + \Delta S^k + \Delta F^k + L + \Delta W}{R}
\end{aligned} \tag{27}$$

The thesis follows from considering that, since the FIFO is emptied and the packet on the link is served at a constant rate R, then $t_c - \overline{t}_c = \frac{\overline{W}(\overline{t}_c) - W(\overline{t}_c)}{R}$.

We never use the link rate to prove the next theorem, stating the B-WFI, as well as the theorem stating the RFI (Appendix B) and the properties these theorems depend on. Hence these theorems hold also for time-varying link rates.

Theorem 2 (B-WFI). *For a flow k, AFQ guarantees*

$$\begin{aligned}
B\text{-}WFI^k = \phi^k \left(\Delta S^k + \Delta F^k\right) + \\
+ (2 - \phi^k) L^k + \phi^k L + \phi^k \Delta W
\end{aligned} \tag{28}$$

PROOF. By substituting (13) in the argument of the *max* function in (11), we get

$$\begin{aligned}
\phi^k W(t_1, t_2) - W^k(t_1, t_2) \overset{(13)}{\leq} \\
\phi^k W(t_1, t_2) - \overline{W}^k(t_1, \max\{t_1, \hat{t}\}) \overset{(21)}{\leq} \\
\phi^k W(t_1, t_2) - \phi^k \left(\overline{W}(\max\{t_1, \hat{t}\}) - W(t_1)\right) + \\
+ \phi^k \left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + L - L^k + \Delta W\right) \leq \\
\phi^k W(t_1, t_2) - \phi^k \left(W(t_2) - W(t_1)\right) + \\
+ \phi^k \left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + L - L^k + \Delta W\right)
\end{aligned} \tag{29}$$

9.1. Discussion of the formulas

Terms related to the scheduling policy. First, the terms proportional to L^k, stemming from Lemma 2, account for the fact that, despite the eligibility constraint, AFQ can be a little ahead of the corresponding ideal system in serving some packets of the flow k. This implies that, during some unlucky time interval, the flow may pay back for the extra service previously received. Second, the terms proportional to L are a consequence of the fact that even the fictitious system (Section 6.3) may serve some packets out of order with respect to finish timestamps (Lemma 5). Finally, the terms proportional to ΔS^k and ΔF^k account for timestamp approximations (Eq. (9)).

Terms related to the FIFO and the uncertainty on $W(t)$. The terms proportional to ΔW embody one of the main results of this paper: with AFQ the FIFO casuses the minimum possible service degradation. In fact, in the T-WFI, the FIFO causes an additional worst-case delay equal to just the time needed to empty the FIFO itself. In terms of B-WFI, the component proportional to ΔW is instead even multiplied by ϕ^k. This is interesting, because low-weight, i.e., low-bandwidth flows (interactive sessions, VoIP, etc.) multiplexed on a high-bandwidth link will not be impacted too badly, in terms of short-term bandwidth guarantees, by the presence of a FIFO even of large size.

10. Comparison with classical bounds and among schedulers

Instantiating the general bounds proved in the previous section, we show and discuss now the service guarantees provided by most AFQ schedulers,

Scheduler	ΔS^k	ΔF^k	Ideal T-WFI	Ideal B-WFI
WF^{2}Q+	0	0	$\frac{L^k}{\phi^k R} + \frac{L-L^k}{R}$	$(1-\phi^k)L^k + \phi^k L$
S-KPS	$4\frac{L^k}{\phi^k}$	$2\frac{L^k}{\phi^k}$	$6\frac{L^k}{\phi^k R} + 2\frac{L}{R}$	$7L^k + 2\phi^k L$
GFQ	$\frac{L^k}{\phi^k}$	$\frac{L^k}{\phi^k}$	$4\frac{L^k}{\phi^k R} + 2\frac{L}{R}$	$5L^k + 2\phi^k L$
QFQ	$2\frac{L^k}{\phi^k}$	$3\frac{L^k}{\phi^k}$	$6\frac{L^k}{\phi^k R} + 2\frac{L}{R}$	$7L^k + 2\phi^k L$
DRR	n.a.	n.a.	$\left(\frac{1}{\phi_{min}} + \frac{1}{\phi^k} + N - 1\right)\frac{L}{R}$	$\left(\frac{\phi^k}{\phi_{min}} + 1 + \phi^k(N-1)\right)L$
Scheduler	ΔS^k	ΔF^k	**Real T-WFI**	**Real B-WFI**
AFQ			$\frac{L^k}{\phi^k R} + \frac{\Delta^k + L - L^k + \Delta W}{R}$	$\phi^k\Delta^k + (2-\phi^k)L^k + \phi^k L + \phi^k\Delta W$
WF^{2}Q+	0	0	$\frac{L^k}{\phi^k R} + \frac{L - L^k + \Delta W}{R}$	$(2-\phi^k)L^k + \phi^k L + \phi^k\Delta W$
S-KPS	$4\frac{L^k}{\phi^k}$	$2\frac{L^k}{\phi^k}$	$7\frac{L^k}{\phi^k R} + \frac{L - L^k + \Delta W}{R}$	$(8-\phi^k)L^k + \phi^k L + \phi^k\Delta W$
GFQ	$\frac{L^k}{\phi^k}$	$\frac{L^k}{\phi^k}$	$3\frac{L^k}{\phi^k R} + \frac{L - L^k + \Delta W}{R}$	$(4-\phi^k)L^k + \phi^k L + \phi^k\Delta W$
QFQ	$2\frac{L^k}{\phi^k}$	$3\frac{L^k}{\phi^k}$	$6\frac{L^k}{\phi^k R} + \frac{L - L^k + \Delta W}{R}$	$(7-\phi^k)L^k + \phi^k L + \phi^k\Delta W$
QFQ+	n.a.	n.a.	$11\frac{L^k}{\phi^k R} + \frac{\Delta W + \min(8L, \Delta W) - L^k}{R}$	$(11-\phi^k)L^k + \min(8L, \Delta W) + \phi^k\Delta W$
DRR	n.a.	n.a.	$\left(\frac{1}{\phi_{min}} + \frac{1}{\phi^k} + N - 1\right)\frac{L}{R} + \frac{\Delta W}{R}$	$\left(\frac{\phi^k}{\phi_{min}} + 1 + \phi^k(N-1)\right)L + \phi^k\Delta W$

Table 2: Service guarantees for various schedulers with "ideal" links (no FIFO, bounds derived from the literature) and "real" links (FIFO, $\overline{W}(t)$, bounds computed from the analysis in this paper). We use the symbol $\Delta^k \equiv \Delta S^k + \Delta F^k$. $0 \leq \phi^k \leq 1$ is the weight of the flow, N is the number of flows, L^k and L are the maximum packet sizes for the flow and among all flows, ΔW is the capacity of the FIFO plus L, and R is the rate of the link.

plus QFQ+ and DRR, in the presence of a FIFO and uncertainties on the work function.

10.1. Ideal and real bounds

Table 2 reports *ideal* (according to classical analysis) and *real* T-WFIs and B-WFIs. For WF^{2}Q+, ideal T/B-WFIs are taken from [1], and are equivalent to our analysis with $\Delta W = 0$, i.e., no FIFO or uncertainty on the work function. However, whereas the real T-WFI of WF^{2}Q+ is correctly equal to the ideal T-WFI plus a term proportional to ΔW, there is a mismatch between ideal and real B-WFIs, because in our analysis we consider a more general time interval (Section 7.2).

The T/B-WFIs for the approximated variants of WF^{2}Q+ in the top half of the table are instead taken from our previous work [3, Sec.6] (replacing σ_i with its value $2\frac{L^k}{\phi^k}$), because not all original papers include all these bounds. These T/B-WFIs actually lie in the middle between ideal and real ones, as they take into account at least the uncertainty on the work function, i.e., assume that schedulers are modified as in Section 6.3 and that $\Delta W = L$. But, with respect to this aspect, the analysis in [3] is less accurate than that in this paper. As a consequence, for the approximated variants of WF^{2}Q+, there are some mismatches between real T/B-WFIs with $\Delta W = L$, in the bottom half of the table, and T/B-WFIs in the top half of the table.

For QFQ+, real T/B-WFIs have been directly computed by the author [16]. We have obtained the bounds in Table 2 by just replacing Q with ΔW in the formulas in [16], and by replacing the resulting bounds with slightly looser, but simpler ones. Finally, the ideal and real T-WFI and B-WFI for DRR can be derived from Eq. (25) with similar steps as in the proofs of theorems 1 and 2.

10.2. Comparison among schedulers

Effect of approximate timestamps in AFQ schedulers. Defined $\Delta^k \equiv \Delta S^k + \Delta F^k$, the bottom half of Table 2 shows that the difference between WF^{2}Q+ and its approximated variants is equal to $\frac{\Delta^k}{R}$ for the T-WFI and $\phi^k\Delta^k$ for the B-WFI. Given the values of Δ^k for these variants, these differences are equal to at most $6\frac{L^k}{\phi^k R}$ and $6L^k$. The following considerations can help put these differences into context.

Tightness of the bounds in practical terms. As thoroughly discussed in [16], most time-sensitive network applications tolerate a maximum packet delay/jitter in the order of the time to transmit one packet at the rate guaranteed to the flows of the application, i.e., in the order of $\frac{L^k}{\phi^k R}$. Similarly, they tolerate a service lag in the order of the size

L^k of the application packets. Therefore, thanks to their T-WFIs and B-WFIs, approximated variants of WF^{2}Q+, as well as QFQ+, are about as effective as WF^{2}Q+ in guaranteeing the feasibility of most time-sensitive applications.

The same property definitely does not hold for round-robin schedulers, such as DRR, whose T-WFIs and B-WFIs contain components proportional to $\frac{L}{\phi_{min}}$ and NL. Then these schedulers may easily cause such a high delay, jitter and lag to make a time-sensitive application unfeasible (Section 7).

Effect of the FIFO. Unfortunately, as ΔW grows, also AFQ schedulers start to suffer from the same problem. In particular, the difference between DRR and an AFQ scheduler becomes negligible if the number of slots in the FIFO is in the order of $N+\frac{1}{\phi_{min}}$. Which is then the best scheduler for such scenarios?

The answer depends mainly on the *cost* of the schedulers on the target system, in terms of running time and software or hardware resources. In this respect, all AFQ schedulers in Table 2 are more complex and have usually a higher running time than DRR. QFQ+ has instead proved to be at least about as fast as DRR on the real systems considered in [16], and, in particular, even faster for the most relevant scenarios.

On systems on which QFQ+ does happen to be at least about as efficient as DRR, QFQ+ may be the best option with large FIFOs. Actually on such systems, QFQ+ is likely to be the best option, in practical terms, for all queue sizes.

On the opposite end, for systems with large FIFOs and on which DRR is the simplest and fastest scheduler, the latter is most likely the best option. In any case, the size of the FIFO may dramatically influence both the absolute and the relative performance of the schedulers. Hence we complete this discussion by reporting typical FIFO sizes.

10.3. FIFO sizes in practical systems

The two queues typically used by a NIC, namely the transmit ring and the hardware FIFO (Section 6.1), operate on largely different timescales. The hardware FIFO should just absorb delays in the order of a few microseconds, which may occur when there is heavy contention on the system bus. Nevertheless, at 10..40 Gbit/s speeds, 10 μs correspond to up to 50 Kbytes of data, which is a reason why the hardware FIFO can be much larger, sometimes up to 512 Kbytes [6].

As for the transmit ring, with packet rates in the range of millions of packets per second, it is hard for the operating system to handle one interrupt per packet. To bound interrupt frequency, most modern NICs and operating systems implement a feature called interrupt coalescing/mitigation, with latencies typically in the 20..100 μs range. On top of this, interrupt handlers might be further delayed because of the CPU being busy with other processes. This motivates the use of extremely large transmit rings. Common values are 64..256 packets for low-speed hardware, and up to 1024..4096 buffers for 10 Gbit/s links. Even if systems are moving to enforce queue limits in terms of bytes [5] and not maximum-size packets, the need to deal with those large latencies means that, especially on high speed links, ΔW is often in the order of $100L$ and more.

11. Conclusions

We have proved analytically the actual service properties of a large family of fair-queueing schedulers, ranging from WF^{2}Q+ to its fast, approximated variants, and of basic round-robin schedulers, in the presence of output FIFOs between the scheduler and the actual link, and without exact knowledge of the number of bits transmitted.

Our main general result is that, with all the schedulers considered in this paper, the degradation of the service guarantees caused by FIFOs consists only in an additional worst-case delay equal to the time needed to empty the FIFOs themselves. The scheduling perturbations induced by these queues may however render completely irrelevant the difference in service properties among different fair-queueing or round-robin schedulers.

These results are of practical relevance because output FIFOs exist in all hardware and software components, and it is important to know their impact when designing systems with tight service guarantees.

Appendices

A. Proofs of lemmas 1 and 2

To prove Lemma 1, we need to prove first the following preliminary relation between the completion time of packet transmissions in the fictitious packet system (Section 6.3) and in its corresponding fluid system (Figure 3 C.). The proof of the relation is

an extension, for the fictitious system and its special link, of the classical proof of the worst-case delay guaranteed by a timestamp-based fair-queueing scheduler [9, 2, 1].

Lemma 5. *Let p_m be the $m-th$ packet served in the fictitious packet system, and t^p_m and t^f_m the completion times of p_m in the fictitious packet system and in its corresponding fluid system (Figure 3 C). If $\overline{F}^k_m = \tilde{F}^k_m$ then $t^p_m \leq t^f_m + \Delta T_L$, where ΔT_L is such that $\overline{W}(t^f_m, t^f_m + \Delta T_L) = L$ (if the exact and the approximate virtual finish time of packet p_m are equal, then p_m will complete in the packet system no later than in the fluid one, plus a worst-case delay equal to ΔT_L).*

Proof. To prove the thesis we prove first that, if all packets are served in finish-time order, then $t^p_m \leq t^f_m$ trivially holds. Otherwise, if some packet is served out-of-order before p_m, then the in-order packets (including p_m) served after the last out-of-order one must have necessarily arrived so late that $t^p_m - t^f_m$ cannot be higher than ΔT_L.

Let o be the smallest index for which all packets have an approximated finish time no greater than p_m, $\tilde{F}_i \leq \overline{F}_m \forall i \in [o..m]$. Since $\overline{F}_i \leq \tilde{F}_i$, and the fluid system (using exact timestamps) completes packets in finish time order, all packets $p_o..p_m$ must also be completed not earlier than t^f_m in the fluid system.

If $o = 1$ then from the origin of time the packet system has served only packets $p_1..p_m$, while the fluid system might have already started service for some subsequent packet. Remembering that both systems have the same work function, the fluid system cannot be ahead of the packet system, hence $t^p_m \leq t^f_m$.

If $o > 1$, then packet p_{o-1} has a higher finish time than $p_o \ldots p_m$, none of which has started in the packet system before t^p_{o-1}. This means that at t^p_{o-1} either they had not yet arrived, or they were not eligible ($\tilde{S}^k(t^P_{o-1}) > \overline{V}(t^P_{o-1})$), so even the fluid system cannot have started serving any of those packets before t^p_{o-1} (the fluid system starts to serve a flow at time t only if $\overline{S}^k(t) \leq \overline{V}(t)$, and $\overline{S}^k(t) \geq \tilde{S}^k(t)$ holds). However, some of the packets $p_o \ldots p_m$ may become eligible while the fluid system works at time t^p_{o-1}, and hence also these packets may receive some service in the fluid system at time t^P_{o-1}. Besides, the work function of the fluid system increases by at most L at time t^p_{o-1}. As a consequence, between t^p_o and t^p_m the fluid system must have done at least the same amount of work as the packet system, minus at most L. Hence t^f_m cannot precede t^p_m by more than ΔT_L.

Proof (Lemma 1). Let $\overline{t}$ be a generic time instant at which flow k is backlogged, with the packet p_m at its head, or just transmitted by the fictitious system if $\overline{t}$ is equal to the transmission time t^p_m of p_m itself in the fictitious system. We know that $\overline{F}^k(\overline{t}) \leq \tilde{F}^k(\overline{t})$. If $\overline{F}^k(\overline{t}) = \tilde{F}^k(\overline{t})$, Lemma 5 tells us that the transmission completion times in the packet and fluid systems are $t^p_m \leq t^f_m + \Delta T_L$. Besides, denoted as $\overline{V}^k_{fluid}(t)$ the virtual time of flow k in the fluid system, the latter guarantees that $V(t) \leq \overline{V}^k_{fluid}(t)$ holds at all times. Thus

$$\overline{V}(\overline{t}) \leq \overline{V}^k_{fluid}(\overline{t}) \overset{\overline{t} \leq t^p_m}{\leq} \overline{V}^k_{fluid}(t^p_m) \overset{t^p_m \leq t^f_m + \Delta T_L}{\leq} \overline{V}^k(t^p_m) + L \overset{(8)+(9)}{\leq} \tilde{F}^k(t^{p-}_m) + L \overset{t^{p-}_m \leq \overline{t}^-}{\leq} \tilde{F}^k(\overline{t}^-) + L \tag{A.1}$$

The case $\overline{F}^k(\overline{t}) < \tilde{F}^k(\overline{t})$ can be handled by considering what happens if packet p_m is artificially extended so $\overline{F}^k(\overline{t}) = \tilde{F}^k(\overline{t})$. The larger packet would still satisfy (A.1). Besides, whether or not the original packet p_m is replaced with a larger one, the values of $\overline{V}(\overline{t})$ and $F^k(\overline{t})$ are the same, because 1) the value of t^p_m does not depend on the size of p_m, 2) $\overline{t} \leq t^p_m$, and 3) the size of p_m does not influence either any timestamp or the packet service order up to time t^p_m. Hence the thesis holds also in this case.

Proof (Lemma 2). Given any time instant t, we consider the smallest time instant t_p such that $\overline{S}^k(t_p) = \overline{S}^k(t)$, and we denote as p_m the packet served at time t_p, and l_m its size. According to (8), either $\overline{S}^k(t_p) = \overline{V}(t_p) \leq \overline{V}(t)$ or $\overline{S}^k(t_p) = \overline{F}^k(t^-_p)$. In the first case the thesis holds trivially. For the thesis to hold in the other case, at least one packet of flow k must have been already served before time t_p. Let t'_p be the largest time instant, with $t'_p < t_p$, at which a packet of flow k is served. Flow k has to be eligible at time t'_p, i.e., $\tilde{S}^k(t^-_p) = \tilde{S}^k(t'_p) \leq \overline{V}(t'_p) \leq V(t^-_p)$ has to hold. Besides, we can note that the virtual time advances by at least the size of p_m at time t_p, thus $\overline{V}(t^-_p) \leq \overline{V}(t_p) - l_m \leq \overline{V}(t) - l_m$ holds. In the end, $\tilde{S}^k(t^-_p) \leq \overline{V}(t) - l_m$. Using this

inequality, we can write

$$\begin{aligned}\overline{S}^k(t_p) = \overline{F}^k(t_p^-) \overset{(8)}{=} \overline{S}^k(t_p^-) + \frac{l_m}{\phi^k} \overset{(4)}{\leq} \\ \tilde{S}^k(t_p^-) + \Delta S^k + \frac{l_m}{\phi^k} \leq \\ \overline{V}(t) - l_m + \Delta S^k + \frac{l_m}{\phi^k} \overset{0<\phi^k\leq 1}{\leq} \\ \overline{V}(t) - L^k + \Delta S^k + \frac{L^k}{\phi^k}\end{aligned} \tag{A.2}$$

B. Relative Fairness Index of AFQ

The Relative Fairness Index (RFI), introduced in [12], is defined as the maximum difference, over any time interval $[t_1, t_2]$ and pair of flows k and h continuously backlogged during $[t_1, t_2]$, between the normalized service received by to the two flows:

$$\text{RFI} \equiv \max_{\forall h,k,[t_1,t_2]} \left| \frac{W^h(t_1,t_2)}{\phi^h} - \frac{W^k(t_1,t_2)}{\phi^k} \right| \tag{B.1}$$

This metric is useful to determine how evenly a scheduler distributes excess bandwidth in case not all flows are backlogged. In this respect, the B-WFI only identifies if a flow goes below its assigned fair share (something that the RFI does not capture).

Theorem 3 (RFI). *AFQ guarantees*

$$\begin{aligned}RFI = \\ \max_{h,k} \Big\{ \Delta S^{h-} + \Delta S^{h+} + \Delta S^k + \Delta F^k + 2L + \\ + \frac{L^h + \Delta W}{\phi^h} - L^h + \frac{L^k + \Delta W}{\phi^k} - L^k \Big\}\end{aligned} \tag{B.2}$$

Proof. Consider two flows, h and k continuously backlogged during a time interval $[t_1, t_2]$. We can upper-bound the normalized service received by flow h as follows:

$$\begin{aligned}\frac{W^h(t_1,t_2)}{\phi^h} \overset{(6)}{\leq} \frac{\overline{W}^h(t_1,t_2) + \Delta W}{\phi^h} = \\ \overline{V}^h(t_2) - \overline{V}^h(t_1) + \frac{\Delta W}{\phi^h} \leq \\ \overline{S}^h(t_2) - \overline{F}^h(t_1) + \frac{\Delta W}{\phi^h} \overset{(15)}{\leq} \\ \overline{V}(t_2) + \Delta S^{h-} + \frac{L^h}{\phi^h} - L^h - \overline{F}^h(t_1) + \frac{\Delta W}{\phi^h} \overset{(4)}{\leq} \\ \overline{V}(t_2) + \Delta S^{h-} + \frac{L^h}{\phi^h} - L^h - \tilde{F}^h(t_1) + \\ + \Delta S^{h+} + \frac{\Delta W}{\phi^h} \overset{(14)}{\leq} \\ \overline{V}(t_2) + \Delta S^{h-} + \frac{L^h}{\phi^h} - L^h - \overline{V}(t_1) + L + \\ + \Delta S^{h+} + \frac{\Delta W}{\phi^h} = \\ \overline{V}(t_1,t_2) + \Delta S^{h-} + \frac{L^h}{\phi^h} - L^h + L + \\ + \Delta S^{h+} + \frac{\Delta W}{\phi^h}\end{aligned} \tag{B.3}$$

As for the lower bound, we have[8]

$$\begin{aligned}\frac{W^k(t_1,t_2)}{\phi^k} = \frac{W^k(t_2) - W^k(t_1)}{\phi^k} \overset{(12)}{\geq} \\ \frac{W^k(t_2) - \overline{W}^k(t_1)}{\phi^k} \overset{(6)}{\geq} \\ \frac{\overline{W}^k(t_2) - \Delta W - \overline{W}^k(t_1)}{\phi^k} \overset{(16)}{\geq} \\ \overline{V}(t_1,t_2) - \frac{L^k}{\phi^k} - L + \\ + L^k - \Delta S^k - \Delta F^k - \frac{\Delta W}{\phi^k}\end{aligned} \tag{B.4}$$

Substituting (B.3) and (B.4) in (B.1), and taking the maximum over all possible flow pairs, we get the thesis.

[8] Same as in Section 8.3, we could derive tighter bounds by considering that $W^k(t_1,t_2) \geq 0$, but the gain of at most $\Delta W/\Phi^k$ would not change the $1/\Phi^k$ behavior of the RFI at the price of an unreadable expression.

References

[1] J. C. R. Bennet and H. Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Trans. on Networking*, 5(5):675–689, 1997.

[2] J. C. R. Bennett and H. Zhang. WF^{2}Q: Worst-case fair weighted fair queueing. *Proc. of IEEE INFOCOM '96*, pages 120–128, March 1996.

[3] F. Checconi, P. Valente, and L. Rizzo. QFQ: Efficient Packet Scheduling with Tight Bandwidth Distribution Guarantees. *IEEE/ACM Trans. on Networking*, 2013 (to appear).

[4] J. Gettys and K. Nichols. Bufferbloat: dark buffers in the internet. *Commun. ACM*, 55(1):57–65, Jan. 2012.

[5] T. Herbert. bql: Byte queue limits. http://lwn.net/Articles/469652/, November 2011.

[6] Intel. Intel 82599 10 gbe controller: Datasheet. http://www.intel.com/content/www/us/en/ethernet-controllers/82599-10-gbe-controller-datasheet.html, September 2012.

[7] M. Karsten. SI-WF^{2}Q: WF^{2}Q approximation with small constant execution overhead. *Proc. of IEEE INFOCOM 2006*, pages 1–12, April 2006.

[8] M. Karsten. Approximation of generalized processor sharing with stratified interleaved timer wheels. *IEEE/ACM Trans. on Networking*, 18(3):708–721, 2010.

[9] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. on Networking*, 1(3):344–357, June 1993.

[10] L. L. Peterson and B. S. Davie. *Computer Networks - A systems Approach*. Morgan Kaufmann Publishers, 2010.

[11] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. *IEEE/ACM Trans. on Networking*, 4(3):375–385, 1996.

[12] S.J.Golestani. A self-clocked fair queueing scheme for broadband applications. *Proc. of IEEE INFOCOM '94*, pages 636–646, June 1994.

[13] D. C. Stephens, J. C. Bennett, and H. Zhang. Implementing scheduling algorithms in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 17(6):1145–1158, June 1999.

[14] D. Stiliadis and A. Varma. A general methodology for designing efficient traffic scheduling and shaping algorithms. *Proc. of IEEE INFOCOM '97*, pages 326–335, 1997.

[15] P. Valente. Exact gps simulation and optimal fair scheduling with logarithmic complexity. *IEEE/ACM Trans. on Networking*, 15(6):1454–1466, 2007.

[16] P. Valente. Reducing the execution time of fair-queueing packet schedulers. *Computer Communications*, 47(0):16 – 33, 2014.

[17] J. Xu and R. J. Lipton. On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms. *IEEE/ACM Trans. on Networking*, 13(1):15–28, 2005.

[18] X. Yuan and Z. Duan. Fair round-robin: A low complexity packet scheduler with proportional and worst-case fairness. *IEEE Trans. on Computers*, 58(3):365–379, 2009.