Analysis of Fair Queueing schedulers in Real Systems

Luigi Rizzo
Universita‘ di Pisa

Paolo Valente
Universita‘ di Modena e Reggio Emilia

Abstract

Research on fair queueing schedulers (WF^{2}Q, WF^{2}Q+ and approximated variants) has led to the design of very efficient algorithms, whose performance ranges from optimal fairness and delay guarantees at $O(\log N)$ cost, to quasi-optimal guarantees at $O(1)$ cost.

These guarantees have been computed assuming that the number of bits transmitted by the communication link is known at any time instant, and that the scheduler is directly attached to the communication link with no interposed buffering.

Both assumptions are unfortunately unrealistic.

Real communication devices normally include FIFOs, possibly very deep ones, between the scheduler and the transmit unit, to avoid that the links becomes idle while the next packet to transmit is selected. The presence of the FIFO has a huge effect on the service guarantees of the scheduler, introducing extra error terms that may be significantly larger than those computed ignoring buffering.

In this paper we make two contributions: i) provide analytical bounds of the effect of FIFOs on the service guarantees of fair queueing schedulers, and ii) put the otherwise dry formulas into practical scenarios, by instantiating the various parameters with realistic values, and comparing several schedulers in presence of variable FIFO size, packet sizes and flow weights. These results should help designers to make informed decisions and sound tradeoff when building systems.

1. INTRODUCTION

Research on packet scheduling has explored many theoretical and practical aspects related to the design and implementation of these systems. Several service metrics such as the Relative Fairness [8] and Worst-case Fairness [2] have been defined to determine the quality of a schedulers. We know how closely a packet-based scheduler can emulate an ideal, fluid system [7] providing perfect bandwidth sharing, and the complexity bounds involved in such emulation [12]. Optimal algorithms matching these lower bounds have been defined [11]. For high speed schedulers, when even the $O(\log N)$ cost in the number of flows is considered too high, there have been several proposal with $O(1)$ complexity and only constant deviation from the optimal schedulers [6, 9, 4]. One of them, QFQ [4] has a speed comparable to robin schedulers while giving the same quasi-optimal service guarantees of the best fair queueing schedulers. It is also readily available in major operating systems, either natively (FreeBSD, Linux) or as part of the dummynet traffic shaper [3].

An element is missing in the picture though. The analysis of packet schedulers is normally done assuming that the communication device can report the exact number of bits transmitted at any time, and is directly attached to the scheduler with no interposed buffering. None of these assumptions holds in practice. First, network interfaces operate on a packet-by-packet basis, and do not export a real-time indication of the number of bits transmitted. Second, communication links, especially the 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.. The existence of the FIFO prevents the link from becoming idle while accessing the next packet to be transmitted, but its implicit principle of operation is that the link appears as having a service rate which is bursty and sometimes much higher than what it is really capable of.

The amount of buffer can be large, even in the order of tens of packets on very fast links, and the device is capable of absorbing large batches of packets at once. This causes two phenomena, which this paper is addressing.

First, the fact that a link may absorb large bursts in very short intervals may cause traffic to be subject to a delay which is, intuitively, at least as large as the FIFO size. Second, the extra delay with respect to the non-FIFO case can even exceed the size of the FIFO, due to the way the scheduler is implemented. The phenomenon is relevant especially in high speed links, where FIFOs can be significantly large.

In this paper we consider a class of fair queueing schedulers that include WF^{2}Q+ and its approximated variants (this class has been defined for the first time in [6]), and make the following contributions: i) we propose a simple modification to these schedulers to approximate the number of bits transmitted, and we provide an analysis of the guarantees that these schedulers provide if they are modified as suggested and if they are followed by a FIFO before the actual communication link; ii) we instantiate the result for WF^{2}Q+ and its approximated variants. To the best of our knowledge, there are no published results on either of the two items. Furthermore, the second contribution is very important in practice because it helps designing systems (choosing algorithms and FIFO sizes) to match the required speed of operation and service guarantees.

The rest of the paper is structured as follows. Section 2 provides a description of the problem that we are addressing, with an example to clarify how the issue has practical consequences. Section 3 briefly describes related work. Section 4 defines the terms used in the paper, and some background on timestamp-based packet schedulers. Readers familiar with the topic may skim over this part, but please refer to it to make sure that there is no ambiguity on the system model and definitions used here (there are some differences in the literature). The core of the paper starts with Section 4.4 where we define a realistic model of a link scheduler with output FIFO, and summarize our proof technique. Supporting lemmas are presented in Section 5, whereas Section 6 contains the proofs of our main results and a discussion of their implications.

2. DESCRIPTION OF THE PROBLEM

We describe our problem with an example, shown in Figure 1. A weighted-fair-queueing scheduler is in charge of arbitrating access to a shared link for a number of flows. The scheduler assigns the link to flows proportionally to a parameter ϕ^k, called the flow's weight.

The ideal, infinitely precise subdivision of the link's capacity (often called "fluid scheduling") is approximated on a packet-by-packet basis, trying to serve flows as close as possible to what would happen in a fluid system. To this end, the scheduler[1] marks a packet i with a "virtual" start time s_i (think of it as derived from a "virtual clock" advancing with the actual number of bits transmitted by the link, and a virtual finish time $f_i = s_i + l_i/\phi_k$ where l_i is the length of the packet. Whenever the link becomes idle, the scheduler selects for transmission the packet with minimum f_i (under some constraints), and then waits until the transmission is complete to select the next one.

[1]This is a simplified (but correct for the example at hand) description of how a fair queueing scheduler works. A complete definition of the algorithm will be presented in Section 4.2.

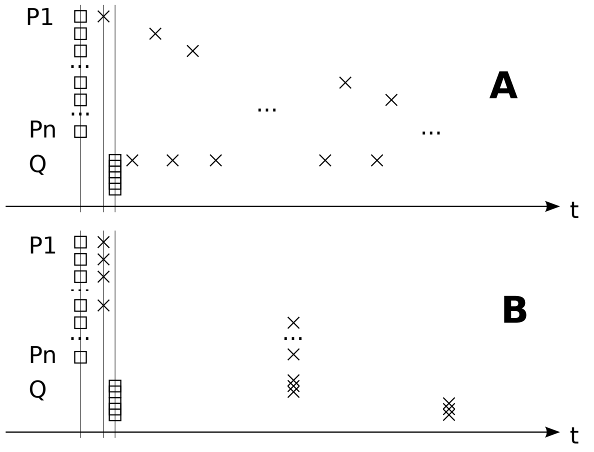

Figure 1: The example in Section 2: squares represent packet arrivals, crosses are departures, flow Q has a much higher weight than the P_i's. Case A: an ideal link without fifo dequeues one packet at a time, insuring a prompt and smooth service to flow Q. Case B: a scheduler is followed by a deep FIFO. A large number of low-weight flows are served immediately, delaying service to Q until the FIFO runs dry.

Assume a system where at some initial time a set of packets arrives at the scheduler for flows $P_1..P_N$, all with the same weight ϕ_1. Shortly afterwards, the scheduler receives a request for new transmission, and after another short interval a set of packets arrive for flow Q, which has a much higher weight than the other flows (for simplicity assume all packets have length L). All packets arriving initially will be marked with the same $< s_0, s_0 + L/\phi_0 >$ pair, as the virtual clock does not advance during the arrival of the burst.

Figure 1-A shows the case of a scheduler driving an ideal link. P_1 first packet is served first, but when the scheduler is invoked next it will start serving packets from Q according to the weights of the backlogged flows. Figure 2-A shows the virtual time markings corresponding to this case.

Conversely, Figure 1-B represents the evolution of the system with a deep FIFO between the scheduler and the link. After the arrival of the initial packets, the scheduler is requested to refill the FIFO at once. The subsequent packets for Q will then have to wait a long time before being dequeued. Also, the timestamps of Q's packets will be much higher (see Figure 2-B) than in the other case due to the initial burst of packets served. Eventually the correct rate will be granted to Q, but only after a possibly large initial delay.

In contrast, in the idealized system without a FIFO between the scheduler and the link, the packet for P_1 will be served immediately, but the virtual clock will only advance by a modest amount, and this will permit

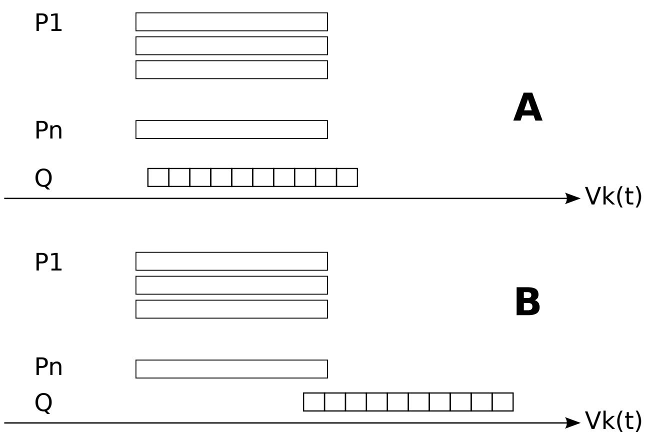

Figure 2: Same example as in Figure 1, this time shown in terms of packet timestamps. In case B, the immediate service of many low-weight packets makes the start time for Q's packets much higher than in case A.

serving Q with the proper rate (in the example, half of the total link capacity) almost right away.

As a result, with the same sequence of (real time) arrivals to the scheduler, and a link with the same capacity, packet Q might be subject to a much higher delay, possibly even exceeding the size of the FIFO.

The purpose of this paper is to evaluate how large is the additional delay, and how it is affected by the FIFO size and the weights of the flows. Considering that the tradeoffs to build efficient ($O(1)$ instead of $O(\log N)$ time) schedulers is paid in terms of a small reordering of output packets, it is important to understand how the effect of the FIFO compares to the previous approximation.

Note that the example given above 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). Similarly, communication hardware (especially high speed NICs) have large output queues or internal FIFOs to make sure that the link does not starve between subsequent packet transmissions.

3. RELATED WORK

The analysis of packet scheduling starts with the seminal work by Parekh and Gallager [7] 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 bounded delay with respect to GPS, also introducing a metric similar to the T-WFI. Followup works include WF^{2}Q+, 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 [11] proved that also WF^{2}Q can be implemented in $O(\log N)$ time). Xu and Lipton [12] 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 [6, 9, 4], presenting $O(1)$ schedulers with quasi-optimal service guarantees. Constant-time fair queueing schedulers have also been built starting from Round-Robin schedulers, some of which [13] integrate concepts used in virtual-time based schedulers to achieve optimal service guarantees.

With only one exception, the analysis in all the papers cited so far has been done in the assumption that the number of bits transmitted is known at any time instant and that any queueing occurs *in the scheduler*: once a packet leaves the scheduler, it immediately starts being transmitted on the output link.

To our knowledge, there is only one recent work [5] that still carries out analysis assuming that the number of bits transmitted is known, but that does account for output buffering. The latter is considered because the scheduler proposed in this work has a large worst-case per-packet execution time, and an output buffer could absorb part of this time. The analysis in [5] is however limited to the evaluation of the time that a packet spends in the buffer, and does not address the impact on the service properties of the scheduler.

4. DEFINITIONS AND SYSTEM MODEL

In this Section we define the concepts and symbols used in the scheduling literature, which sometimes uses different notations for the same concepts. For convenience, all symbols used in this paper are listed in Table 1.

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 $f(t_0^-) = \lim_{\epsilon \to 0} f(t_0 - |\epsilon|)$.

4.1 System model

Consider a system as in Figure 3, in which N packet flows (defined in whatever meaningful way) share a common transmission link serving one packet at a time. The link has a time-varying rate, with $W(t)$ being its "work function", or the total number of bits transmitted in $[0, t]$. A system is called *work conserving* if the link is used at full capacity whenever there are packets queued.

Symbol	Meaning
$*^k$	A superscript indicates a quantity related to a flow
$*_m$	A subscript indicates a quantity related to a packet
N	Total number of flows
h, k	Flow index
L, L^k	Max length of any packet in the system/flow
ϕ^k	Weight of flow k
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 4.3
$\overline{W}(t), \overline{W}^k(t)$	The "work function" describing the input to the FIFO
$\overline{V}(t), \overline{V}^k(t)$	System/flow virtual time corresponding to $\overline{W}(t)$
$\overline{S}^k, \overline{F}^k$	Counterparts of S^k and F^k obtained using $\overline{V}(t)$ instead of $V(t)$ in (2)
$\tilde{S}^k, \tilde{F}^k$	Counterparts of $\hat{S}^k$ and $\hat{F}^k$ obtained using $\overline{V}(t)$ instead of $V(t)$ in Eq. (2)
$B(t)$	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.

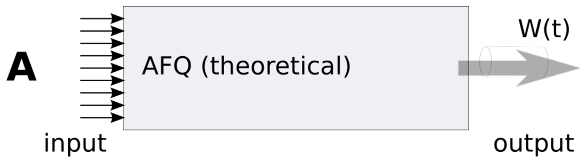

Figure 3: The system model used in the literature: a scheduler (AFQ) drives directly an ideal link with work function $W(t)$.

A scheduler (the AFQ block in the figure) 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.

In our model, each flow k is assigned a fixed weight $\phi^k > 0$. Without losing generality, we assume that $\phi = \sum_{k=1}^{N} \phi^k \leq 1$.

A flow is defined *backlogged/idle* if it owns/does not own packets not yet completely transmitted. We call $B(t)$ the set of flows backlogged at time t. Each flow uses a FIFO queue to hold the flow's own backlog.

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.

The analysis of the schedulers considered in this paper uses the concept of *corresponding systems* [2, Definition 1]: two systems are corresponding if they have the same work function $W(t)$, serve the same set of flows with the same weights in both systems, and are subject to the arrival pattern.

4.2 WF^{2}Q+

The schedulers we consider are approximated variants of the WF^{2}Q+ algorithm, and we call them AFQ (Approximated Fair Queueing) schedulers.

Here we outline the WF^{2}Q+ algorithm for the case of a variable-rate link (see [1, 10] for a complete description). WF^{2}Q+ is a *packet scheduler* that approximates, on a packet-by-packet basis, the service provided by a corresponding work-conserving *ideal fluid system* that delivers the following, almost perfect bandwidth distribution over any time interval during which a flow is continuously backlogged:

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

The fluid and the packet system differ in that the former may serve multiple packets in parallel, whereas the latter has to serve one packet at a time, and is non preemptive. To define the scheduling policy of WF^{2}Q+, we need to introduce the concept of *eligibility*, first defined in [2, Section 3]: 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+ operates as follows. Each time the link is ready, the scheduler starts to serve, among the eligible packets, the next one that would be completed in the fluid system; ties are arbitrarily broken. WF^{2}Q+ is a work-conserving on-line algorithm, hence it succeeds in finishing packets in the same order as the ideal fluid system, except when the next packet to serve arrives after one or more out-of-order packets have already started.

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., 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[2] associated with a virtual time $V^k_{fluid}(t)$ also in the fluid sys-

[2]this parameter is not needed in the implementation but we use it to prove Lemma 2.

tem. $V^k_{fluid}(t)$ is computed with the same rules as $V^k(t)$, but, of course, since the amount of service received by a flow in the packet system scheduled by WF^{2}Q+ may differ with that in the corresponding fluid system, the values of $V^k(t)$ and $V^k_{fluid}(t)$ may differ.

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 in the fluid system. Using this definition, we define the *virtual start* and *finish time* 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. On these events $S^k(t)$ and $F^k(t)$ are updated as follows:

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

where t_p is the time when a packet enqueue/dequeue occurs, and l^k is the packet size. $V(t)$ is the *system virtual time* function defined as follows (assuming $\sum \phi^k \le 1$):

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

Note that the instantaneous link rate needs not be known to update $V(t)$, and just $W(t_1,t_2)$ (the amount of data transferred in $[t_1,t_2]$) suffices. At system start up, $V(0) = 0$, $S^k(0) \leftarrow 0$ and $F^k(0) \leftarrow 0$. The scheduling policy of WF^{2}Q+ is implemented using only $V(t)$, and the virtual start and finish times of the flows, as detailed in the next paragraph.

4.2.1 Scheduling decisions

In terms of virtual times, flow k is *eligible* at time t if $V(t) \ge S^k(t)$. In addition, the fluid system serves flows so as to complete packets in virtual finish time order [1, 10]. WF^{2}Q+ can then be implemented as follows: 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.

4.3 Approximated variants of WF^{2}Q+

The exact WF^{2}Q+ algorithm as described above, has $\Omega(\log N)$ complexity in the number of flows [12]. In order to implement the same policy in $O(1)$ time, several schedulers [6, 9, 4] label flows with approximated virtual start and finish times $\hat{S}^k(t)$ and $\hat{F}^k(t)$, in addition to the exact values defined in (2). Lowest-cost examples are QFQ [4], S-KPS [6] and the scheduler proposed in [9], which we call GFQ hereafter. The approximated values help reducing the complexity of certain sorting stages in the algorithm, making them constant-time operations.

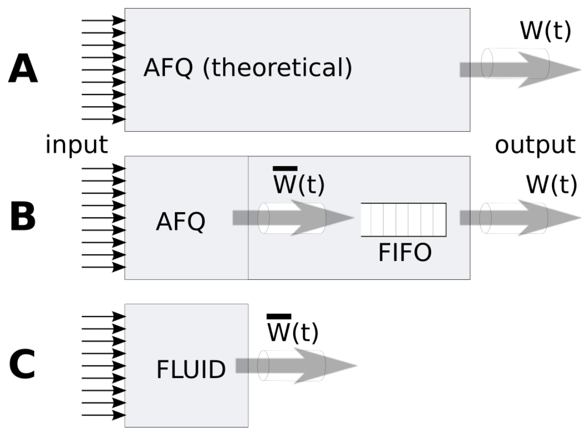

Figure 4: A: the system model used in the literature, ignoring the presence of the FIFO and assuming $W(t)$ is known exactly within the scheduler. B: a real system, made of an AFQ scheduler feeding a dequeue unit with work function $\overline{W}(t)$, followed by a FIFO and the output link. C: the corresponding fluid system for the first part of B), serving multiple packets at a time, with the same work function $\overline{W}(t)$.

The way approximations are computed varies, but in all cases we have:

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

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.

AFQ uses the approximated timestamps to compute the virtual time, i.e., it uses $\hat{S}^k(t_2)$ instead of $S^k(t_2)$ in (3), and to choose the next packet to transmit (Sec 4.2.1), while it uses the exact timestamps to charge flows for the work received (Eq. (2)).

4.4 Introducing an output queue

As mentioned, the service guarantees of packet schedulers are generally computed on the model in Figure 3, which assumes that i) the exact value of $W(t)$ is known when updating $V(t)$, and ii) the link requests a new packet to transmit only once the previous one has been fully transmitted.

These two assumptions are almost never true in a real system.

First, network interfaces (commonly called *NICs*) operate on a packet-by-packet basis, and do not export a real-time indication of the number of bits transmitted. Even the notification of transmission completions, avail-

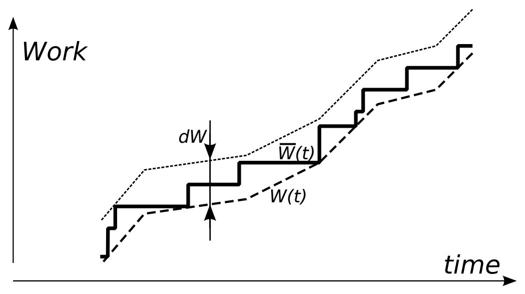

Figure 5: $\overline{W}(t)$ is within a band of height ΔW above $W(t)$.

able through memory mapped registers or interrupts, can be delayed by several microseconds, corresponding to tens/hundreds of packets on high speed links. Secondly, to make sure that the link does not remain idle while the scheduler (firmware or software) provides the next packet to send, NICs typically implement i) a large ring buffer, where the operating system can push outgoing transmissions, and ii) an internal FIFO that drains the ring buffer and drives the link.

A realistic model of a communication device is thus the one in Figure 4-B, where the scheduler is drained by a dequeue unit that takes care of inserting packets in the FIFO queue. We denote as $\overline{W}(t)$ the sum of the sizes of the packets 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. As shown in Figure 5, the work function $\overline{W}(t)$ has a stepwise shape and lies in a band of height ΔW on top of $W(t)$, where ΔW equals the maximum capacity of the FIFO plus the size of the packet under service on the link.

Filling and draining the ring and the FIFO typically occurs with some hysteresis, so that the buffers are filled until a high water mark is reached, then drained until a low water mark or a sufficient amount of time have elapsed, and so on. In formula, we have

$$\Delta W = \max_t \overline{W}(t) - W(t) \geq 0. \tag{5}$$

The presence of the FIFO, and the different work functions used to drain the scheduler may change the order in which packets are served between the ideal (Figure 4-A) and the actual (Figure 4-B) systems. Section 2 gives one example of what can happen, but there are many other pathological patterns that may occur, affecting the service guarantees of the overall system.

4.5 Our contribution

As a consequence, our goal, and the main contribution of this paper, is to evaluate the service properties of systems modeled as in Figure 4-B, hence taking into account the impact of FIFOs and the uncertainty on the work function $W(t)$. The goal is achieved in three steps:

1. We first prove a few preliminary lemmas (Section 5) that are needed in the proofs of the main theorems in this paper.
   Readers not interested in the details of the subsequent proofs may skip this section.

2. We then prove our main results (Section 6), namely the various fairness indexes (B-WFI, RFI and T-WFI) for a generic AFQ scheduler in presence of a FIFO and approximated work function. These equations are mostly mechanical derivations from the definitions and the previous lemmas. We also extend these (sometimes hard to read) results with approximated (and coarser) bounds trying to reduce the huge number of terms in these equations.

3. Finally, we compare these results with those derived in the literature for various scheduling algorithms, both with and without FIFOs. Also, we provide realistic estimates of the service guarantees using values (queue and packet sizes, data rates, etc.) from actual hardware and software.

4.6 Proof machinery

We compute the B-WFI and the RFI by lower-bounding the amount of service $W^k(t_1, t_2)$ given by the system in Figure 4-B to a flow during any time interval in which the flow is continuously backlogged in the scheduler (left part of Figure 4-B)[3]. To lower-bound $W^k(t_1, t_2)$ we model our system as the cascade shown in Figure 4-B. The scheduler uses $\overline{W}(t)$ as the work function, and Equations 2 and 3 to compute the virtual time $\overline{V}(t)$ and the timestamps $\overline{S}^k(t)$ and $\overline{F}^k(t)$. We call $\tilde{S}^k(t)$ and $\tilde{F}^k(t)$ the approximate versions of the flows' timestamps. The virtual time $\overline{V}(t)$ then becomes

$$\overline{V}(t_2) \equiv \max\left\{\overline{V}(t_1) + \overline{W}(t_1, t_2), \min_{k \in B(t_2)} \tilde{S}^k(t_2)\right\} \tag{6}$$

and the relation between timestamps is

$$\begin{aligned} \overline{S}^k(t) - \Delta S^k \leq \tilde{S}^k(t) \leq \overline{S}^k(t) \leq \\ \overline{F}^k(t) \leq \tilde{F}^k(t) \leq \overline{F}^k(t) + \Delta F^k \end{aligned} \tag{7}$$

For some derivations, we compare the behaviour of the (packet) AFQ scheduler on the left of Figure 4-B, with its *corresponding fluid system* shown in Figure 4-C. The two have the same input pattern and the same work function $\overline{W}(t)$, but the packet systems serves one packet at a time whereas the fluid system may serve multiple flows in parallel.

While it may seem counterintuitive, it is easier to compute a lower bound to $\overline{W}^k(t_1, t_2)$, the amount of work received by a flow on entry to the FIFO during

[3]The quantity of interest is $W^k(t_1, t_2)$, i.e. packets actually exiting the link, and not $\overline{W}^k(t_1, t_2)$, which only refers to packets entering the FIFO.

$[t_1, t_2]$, than a lower bound to $W^k(t_1, t_2)$. In this respect, in Section 5.2 we compute two slightly different bounds to $\overline{W}^k(t_1, t_2)$. From the first bound we immediately get the T-WFI. Using the second bound we compute instead the B-WFI and the T-WFI by a few simple steps. Especially, we achieve the latter result thanks to the following reduction.

4.7 Problem reduction

The bound on $W^k(t_1, t_2)$ can be derived by one on $\overline{W}^k(t_1, \hat{t})$, the amount of work received by a flow on entry to the FIFO in a suitable interval $[t_1, \hat{t}]$. To this purpose, let $I = [\hat{t}, t_2]$ be the interval such that all packets that are in the FIFO or under service[4] at time t_2 have been dequeued during I. By definition, the sum of the sizes of all these packets is $\overline{W}^k(\hat{t}, t_2)$, and we have

$$0 \le \overline{W}^k(t_2) - W^k(t_2) \le \overline{W}^k(\hat{t}, t_2) \tag{8}$$

Using these two inequalities, we can then write:

$$\begin{aligned} W^k(t_1, t_2) = W^k(t_2) - W^k(t_1) \ge \\ \max\{0, W^k(t_2) - \overline{W}^k(t_1)\} \ge \\ \max\{0, \overline{W}^k(t_2) - \overline{W}^k(\hat{t}, t_2) - \overline{W}^k(t_1)\} = \\ \max\{0, \overline{W}^k(t_2) - \overline{W}^k(t_2) + \overline{W}^k(\hat{t}) - \overline{W}^k(t_1)\} = \\ \max\{0, \overline{W}^k(\hat{t}) - \overline{W}^k(t_1)\} = \\ \overline{W}^k(t_1, \max\{t_1, \hat{t}\}) \end{aligned} \tag{9}$$

Intuitively, this equation relates the output service in $[t_1, t_2]$ with the service given to the scheduler during a shorter interval which excludes packets still staged in the FIFO or in the link.

In Section 5.2, Lemma 5 we compute an upper bound to $\overline{W}^k(t_1, t_2)$ for any time interval $[t_1, t_2]$ during which flow k is continuously backlogged. In case $\hat{t} > t_1$, from this bound we get immediately an upper bound to $\overline{W}^k(t_1, \hat{t})$ by just replacing t_2 with $\hat{t}$ (in fact, also $[t_1, \hat{t}]$ is a time interval during which flow k is continuously backlogged).

5. SUPPORTING LEMMAS

The following lemmas are intermediate results needed to prove our service-guarantee bounds. We present them in bottom-up order.

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

[4]We will often refer to these packets in the next Sections. These packets have been fully served as far as AFQ is concerned, but have not yet emerged from the link, so from the user's perspective they are not served yet.

We start by comparing the completion time of packet transmissions in the packet and the fluid systems. Let p_m be the $m-$th packet served in the packet system (the two systems do not necessarily complete packets in the same order) and call t^p_m and t^f_m their completion times in the two systems.

LEMMA 1. *If $\overline{F}^k_m = \tilde{F}^k_m$ then $t^p_m \le t^f_m$ (if the exact and the approximated virtual finish time of packet p_m are equal, then p_m will complete in the packet system not later than in the fluid one).*

Intuition for the proof: the packet system serves packets in strict finish time order, except when packets are not eligible or not arrived. For all in-order bursts immediately after an out-of-order packet, the fluid system cannot have started serving any of the packets in the burst before the beginning of the burst in the packet system, so it must finish the burst no earlier than the packet system.

PROOF. Let o be the smallest index for which all packets have an approximated finish time no greater than p_m, $\tilde{F}_i \le \overline{F}_m \forall i \in [o..m]$. Since $\overline{F}_i \le \tilde{F}_i$, 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 $t^p_m \le t^f_m$.

If $o > 1$, then packet p_{o-1} has a higher finish time than $p_o..p_m$, 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 arrived yet, 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 before t^p_{o-1} (the fluid system starts to serve a flow at time t only if $\overline{S}^k(t) \le \overline{V}(t)$ and $\overline{S}^k(t) \ge \tilde{S}^k(t)$ holds). As a consequence, between t^p_{o-1} and t^p_m the fluid system must have done at least the same amount of work as the packet system, hence proving again that t^f_m cannot precede t^p_m.

□

5.1 Globally Bounded Timestamps

The flow timestamps cannot deviate too much from the system's virtual time $\overline{V}(t)$. This is the "Globally Bounded Timestamp" property (GBT) defined in [9, Definition 3]. Here we compute a variant of this property that comes in handy to upper-bound $\overline{W}^k(t_1, t_2)$.

LEMMA 2 (LOWER BOUND FOR $\tilde{F}^k(t)$). *For all times t at which flow k is backlogged,*

$$\tilde{F}^k(t) - \overline{V}(t) \ge 0. \tag{10}$$

Proof. Let $\bar{t}$ be a generic time instant at which flow k is backlogged, with p_m at its head. We know that $\overline{F}^{(}\bar{t}) \le \tilde{F}^k(\bar{t})$. If $\overline{F}^k(\bar{t}) = \tilde{F}^k(\bar{t})$, Lemma 1 tells us that the transmission completion times in the packet and fluid systems are $t_m^P \le t_m^F$. Denoted as $\overline{V}^k_{fluid}(t)$ the virtual time of flow k in the fluid system, the latter guarantees that $V(t) \le \overline{V}^k_{fluid}(t)$ holds at all times. Thus

$$\overline{V}(\bar{t}) \le \overline{V}^k_{fluid}(\bar{t}) \overset{\bar{t}\le t_m^P}{\le} \overline{V}^k_{fluid}(t_m^P) \overset{t_m^P \le t_m^F}{\le} \overline{V}^k(t_m^P) \overset{(2)}{=} \tilde{F}^k(\bar{t}). \tag{11}$$

The case $\overline{F}^k(\bar{t}) < \tilde{F}^k(\bar{t})$ can be handled by considering what happens if packet p_m is artificially extended so $\overline{F}^k(\bar{t}) = \tilde{F}^k(\bar{t})$. The larger packet would still satisfy (11). Besides, whether or not the original packet p_m is replaced with a larger one, the values of $\overline{V}(\bar{t})$ and $F^k(\bar{t})$ are the same, because 1) the value of t_m^p does not depend on the size of p_m, 2) $\bar{t} < t_m^p$, and 3) the size of p_m does not influence either any timestamp or the packet service order up to time t_m^p. Hence the thesis holds also in this case. □

Lemma 3 (Upper bound for $\overline{S}^k(t)$). *At all times t*

$$\overline{S}^k(t) \le \overline{V}(t) + \Delta S^k + \frac{L^k}{\phi^k} - L^k \tag{12}$$

Note: differently from the previous bound, this bound applies to the exact timestamp, as this is what we need in the proof of subsequent Lemma 4.

Proof. 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 (2), either $\overline{S}^k(t_p) = \overline{V}(t_p) \le \overline{V}(t)$ or $\overline{S}^k(t_p) = \overline{F}^k(t_p^-))$. In the first case the thesis holds trivially. For the other case to hold, 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., $\tilde{S}^k(t_p^-) = \tilde{S}^k(t'_p) \le \overline{V}(t'_p) \le V(t_p^-)$ 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 $\overline{V}(t_p^-) \le \overline{V}(t_p) - L_m \le \overline{V}(t) - L_m$ holds. In the end, $\tilde{S}^k(t_p^-) \le \overline{V}(t) - L_m$. Using this inequality, we can write

$$\begin{aligned}\overline{S}^k(t_p) = \overline{F}^k(t_p^-) \overset{(2)}{=} \overline{S}^k(t_p^-) + \frac{L_m}{\phi^k} \overset{(4)}{\le}\\ \tilde{S}^k(t_p^-) + \Delta S^k + \frac{L_m}{\phi^k} \le \overline{V}(t) - L_m + \Delta S^k + \frac{L_m}{\phi^k} \overset{\phi^k\le 1}{\le}\\ \overline{V}(t) - L^k + \Delta S^k + \frac{L^k}{\phi^k}\end{aligned} \tag{13}$$

□

5.2 Lower bounds for $\overline{W}^k(t_1, t_2)$

This section contains lower bounds[5] to $\overline{W}^k(t_1, t_2)$, expressed in terms of the virtual time $\overline{V}(t)$ or the work function $W(t)$. We use the former is used in the derivation of the RFI, wheres we substitute the latter in (9) to compute the B-WFI.

Lemma 4.

$$\begin{aligned}\overline{W}^k(t_1, t_2) \ge\\ \phi^k\overline{V}(t_1, t_2) - \phi^k\left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k - L^k\right)\end{aligned} \tag{14}$$

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 $V^k(t_2) - \overline{V}^k(t_1)$:

$$\begin{aligned}\overline{W}^k(t_1, t_2) =\\ \phi^k\overline{V}^k(t_1, t_2) =\\ \phi^k\left[\overline{V}^k(t_2) - \overline{V}^k(t_1)\right] =\\ \phi^k\left[(\overline{V}(t_2) - \overline{V}(t_1)\right] +\\ \phi^k\left[\overline{V}^k(t_2) - \overline{V}(t_2) - (\overline{V}^k(t_1) - \overline{V}(t_1))\right]\end{aligned} \tag{15}$$

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

$$\begin{aligned}\overline{V}^k(t_2) - \overline{V}(t_2) = \overline{S}^k(t_2) - \overline{V}(t_2) \overset{(2)}{\ge}\\ \overline{F}^k(t_2) - \frac{L^k}{\phi^k} - \overline{V}(t_2) \overset{(4)}{\ge}\\ \tilde{F}^k(t_2) - \Delta F^k - \frac{L^k}{\phi^k} - \overline{V}(t_2) \overset{(10)}{\ge}\\ -\Delta F^k - \frac{L^k}{\phi^k}.\end{aligned} \tag{16}$$

As for the second term, we have

$$\begin{aligned}-\left[\overline{V}^k(t_1) - \overline{V}(t_1)\right] =\\ -\left[\overline{S}^k(t_1) - \overline{V}(t_1)\right] \overset{(12)}{\ge}\\ -\left[\overline{V}(t_1) + \Delta S^k + \frac{L^k}{\phi^k} - L^k\overline{V}(t_1)\right] =\\ -\left[\Delta S^k + \frac{L^k}{\phi^k} - L^k\right]\end{aligned} \tag{17}$$

[5]Remembering that by definition $\overline{W}^k(t_1, t_2) \ge 0$, we could derive tighter bounds by writing $\overline{W}^k(t_1, t_2) \ge \max\{0, ...\}$. 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.

Replacing the two bounds in (15), and rearranging terms, we get the thesis. □

LEMMA 5.

$$\overline{W}^k(t_1,t_2) \geq \phi^k W(t_1,t_2) - \phi^k\left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + \Delta W - L^k\right) \quad (18)$$

PROOF. We prove the thesis by upper-bounding the term $\overline{V}(t_1,t_2)$ in (14) as follows:

$$\begin{aligned} \overline{V}(t_2) - \overline{V}(t_1) &\geq \\ \overline{W}(t_2) - \overline{W}(t_1) &\overset{(5)}{\geq} \\ W(t_2) - \overline{W}(t_1) &\overset{(5)}{\geq} \\ W(t_2) - W(t_1) - \Delta W. \end{aligned} \quad (19)$$

□

6. SERVICE PROPERTIES

We are now ready to compute various service properties of the AFQ family of schedulers, compare the results with the (exact, but theoretical) analysis provided in the literature, and make practical considerations on the impact of the FIFO on the service guarantees.

We focus on three metrics, which express from different points of view the deviation from an ideal behaviour: the Relative Fairness Index (RFI), introduced in [8], the Bit Worst-Case Fairness Index (B-WFI), introduced in [1], and the Time Worst-Case Fairness Index (T-WFI), first introduced in [2].

The theorems stating the B-WFI and the RFI, as well as the properties they depend on, are proven without ever using the link rate. Hence these theorems hold also for time-varying link rates.

6.1 B-WFI

The B-WFI^k for a flow k is defined as:[6]

$$\text{B-WFI}^k \equiv \max_{[t_1,t_2]} \left\{\phi^k W(t_1,\, t_2) - W^k(t_1,\, t_2)\right\} \quad (20)$$

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.

This metric indicates how much, in terms of service received, a flow remains behind what it would receive on an ideal scheduler. The metric is computed over any time interval in which flows are backlogged, so any service oscillation will have a negative impact on the B-WFI^k. Furthermore, if a scheduler does not distribute bandwidth to flows according to their weights, the overall B-WFI $= \max_k \left\{\text{B-WFI}^k\right\}$ will be equally penalized.

THEOREM 1 (B-WFI). *For a flow k, AFQ guarantees*

$$\textit{B-WFI}^k \leq 2\phi^k\Delta W + \phi^k\left(\Delta F^k + \Delta S^k\right) + 2L^k - \phi^k L^k \quad (21)$$

Discussion: The B-WFI is made of three terms. $2\phi^k\Delta W$ accounts for the uncertainty on $W(t)$ and the presence of the FIFO, and it affects high-weight flows more than low-weight ones. This is interesting because, for instance, low bandwidth flows (interactive sessions, VOIP, etc.) multiplexed on a high bandwidth link will not be impacted too badly by the presence of a FIFO even of large size. The term $\phi^k\left(\Delta F^k + \Delta S^k\right)$ accounts for the use of approximate timestamps. In practice (see Table 2) the approximations on timestamps are $\propto 1/\phi^k$ so this gives a constant component, comparable to the final term.

As an example, WF^{2}Q+ uses exact flow timestamps ($\Delta S^k = \Delta F^k = 0$). The use of $\overline{W}(t)$ as work function, without a FIFO makes $\Delta W = L$ (the packet under service). Assuming $L^k = L$, we have B-WFI$^k \leq 2L + \phi^k L$. In the case of QFQ, the approximation on timestamps is $\Delta S^k = \Delta F^k = 2\frac{L^k}{\phi^k}$ so we end up with B-WFI$\approx 6L + \phi^k L$

Conversely, when the FIFO exists and becomes large, say M slots, the term $2\phi^k ML$ becomes the dominant one.

PROOF. By substituting (9) in (20), we get

$$\begin{aligned} &\text{B-WFI}^k \overset{(9)}{\leq} \\ &\phi^k W(t_1,\, t_2) - \overline{W}^k(t_1, \max\{t_1,\hat{t}\}) \overset{(18)}{\leq} \\ &\phi^k W(t_1,\, t_2) - \phi^k W(t_1, \max\{t_1,\hat{t}\}) + \\ &+\phi^k\left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + \Delta W - L^k\right) = \\ &\phi^k W\left(\max\{t_1,\hat{t}\},\, t_2\right) + \\ &+\phi^k\left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + \Delta W - L^k\right) \overset{(5)}{\leq} \\ &\phi^k\Delta W + \phi^k\left(2\frac{L^k}{\phi^k} + \Delta S^k + \Delta F^k + \Delta W - L^k\right) = \\ &2\phi^k\Delta W + \phi^k\left(\Delta F^k + \Delta S^k\right) + 2L^k - \phi^k L^k \end{aligned} \quad (22)$$

□

6.2 T-WFI

[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.

The T-WFI is the counterpart of the B-WFI in terms of delays. Unlike the B-WFI, it can only be computed if the work function $W(t)$ is known. An exact computation is extremely difficult in practice because most links have non-constant, and possibly even load-dependent rate. As an example, wired ethernets have a fixed-size preamble in front of each packet, irrespective of its size; time-multiplexed media serve clients in periodic bursts; radio links may have variable access time (depending on collisions) and data rate (depending on channel conditions).

For simplicity, we make the computation assuming that the link has a constant rate R, i.e. $W(t_1, t_2) = R(t_2 - t_1)$ when there is backlogged traffic. In this case, the T-WFI^k for flow k is defined as

$$\text{T-WFI}^k \equiv \max\left(t_c - t_a - \frac{Q^k(t_a)}{\phi^k R}\right) \tag{23}$$

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.

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

$$\textit{T-WFI}^k \le 2\frac{L^k}{\phi^k R} + \frac{\Delta S^k + \Delta F^k + 2\Delta W - L^k}{R} \tag{24}$$

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 $\bar{t}_c$, then we add to this upper bound the maximum time that may elapse from time $\bar{t}_c$ 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 $\bar{t}_c$, we have that $\overline{W}^k(t_a, \bar{t}_c) = Q^k(t_a)$. Using this equality and (18), and recalling that the link works at constant speed R, we can write

$$\begin{aligned} \bar{t}_c - t_a \le \frac{W(t_a, \bar{t}_c)}{R} &\overset{(18)}{\le} \\ \frac{\overline{W}^k(t_a, \bar{t}_c)}{\phi^k R} + \frac{2\frac{L^k}{\phi^k} - L^k + \Delta S^k + \Delta F^k + \Delta W}{R} &= \\ \frac{Q^k(t_a)}{\phi^k R} + \frac{2\frac{L^k}{\phi^k} - L^k + \Delta S^k + \Delta F^k + \Delta W}{R}&. \end{aligned} \tag{25}$$

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 $t_c - \bar{t}_c \le \frac{\Delta W}{R}$. $\square$

6.3 RFI

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

$$\text{RFI} \equiv \max_{\forall h,k,[t_1,t_2]} \left| \frac{W^h(t_1, t_2)}{\phi^h} - \frac{W^k(t_1, t_2)}{\phi^k} \right| \tag{26}$$

This metric was introduced in [8] and it is useful to determine how evenly a scheduler distributes excess bandwidth in case not all flows are backlogged. In fact, the B-WFI only identifies if a flow goes below its assigned fair share (something that the RFI does not capture).

Theorem 3 (T-WFI). *AFQ guarantees that*

$$\begin{aligned} &\textit{RFI} \le \\ &\max_{h,k} \{\Delta S^h + \Delta F^h + \Delta S^k + \Delta F^k + \\ &+ \frac{L^h + \Delta W}{\phi^h} - L^h + \frac{2L^k + \Delta W}{\phi^k} - L^k\} \end{aligned} \tag{27}$$

Discussion: the presence of the FIFO, and the approximation on timestamps, is more evident in the RFI, as it uses normalized service as a metric. The deviations from the ideal service (which are bounded in absolute values by a small multiple of the FIFO size) are amplified by $1/\phi^k$.

Same as for the B-WFI we have one term accounting for the approximate timestamps (as before, there is an implicit $1/\phi^k$ in all these error terms), and another one with components $\Delta W/\phi^k$ accounting for the effect of the FIFOs.

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:

$$\begin{aligned} \frac{W^h(t_1, t_2)}{\phi^h} &\overset{(5)}{\le} \\ \frac{\overline{W}^h(t_1, t_2) + \Delta W}{\phi^h} &= \\ \overline{V}^h(t_2) - \overline{V}^h(t_1) + \frac{\Delta W}{\phi^h} &\le \\ \overline{S}^h(t_2) - \overline{F}^h(t_1) + \frac{\Delta W}{\phi^h} &\overset{(12)}{\le} \\ \overline{V}(t_2) + \Delta S^h + \frac{L^h}{\phi^h} - L^h - \overline{F}^h(t_1) + \frac{\Delta W}{\phi^h} &\overset{(5)}{\le} \\ \overline{V}(t_2) + \Delta S^h + \frac{L^h}{\phi^h} - L^h - \tilde{F}^h(t_1) + \Delta F^h + \frac{\Delta W}{\phi^h} &\overset{(10)}{\le} \\ \overline{V}(t_2) + \Delta S^h + \frac{L^h}{\phi^h} - L^h - \overline{V}(t_1) + \Delta F^h + \frac{\Delta W}{\phi^h} &= \\ \overline{V}(t_1, t_2) + \Delta S^h + \frac{L^h}{\phi^h} - L^h + \Delta F^h + \frac{\Delta W}{\phi^h}& \end{aligned} \tag{28}$$

As for the lower bound, we have[7]

[7]Same as in Section 5.2, we could derive tighter bounds by

$$\begin{aligned}
\frac{W^k(t_1,t_2)}{\phi^k} &= \\
\frac{W^k(t_2) - W^k(t_1)}{\phi^k} &\overset{(8)}{\geq} \\
\frac{W^k(t_2) - \overline{W}^k(t_1)}{\phi^k} &\overset{(5)}{\geq} \qquad (29)\\
\frac{\overline{W}^k(t_2) - \Delta W - \overline{W}^k(t_1)}{\phi^k} &\overset{(14)}{\geq} \\
\overline{V}(t_1,t_2) - 2\frac{L^k}{\phi^k} + L^k - \Delta S^k - \Delta F^k - \frac{\Delta W}{\phi^k}
\end{aligned}$$

Substituting (28) and (29) in (26), and taking the maximum over all possible flow pairs, we get the thesis. □

7. COMPARISON OF AFQ SCHEDULERS

We can now discuss the behaviour of the AFQ scheduler variants in presence of a FIFO and uncertainties on the work functions.

The various options for schedulers are generally evaluated in terms of computational costs (both asymptotic and actual), and service guarantees. For the latter, the analysis performed with the ideal system model of Figure 3 might be misleading as it misses a component which is sometimes significant. Most of the values are taken from our previous work [4, Sec.6] because not all papers included these results.

Table 2 summarizes the theoretical and actual service guarantees for a few schedulers proposed in the literature (for simplicity we include only the B-WFI and the RFI). For most of the approximated schedulers the results are expressed in terms of the σ_k parameters, which determine how flows are grouped together. In general $\sigma_k = cL^k/\phi^k$ where the constant c is a small integer (1..4). The table shows clearly that the difference between WF^{2}Q+ and the approximated schedulers is just in the component $\phi_k\sigma_i$, which is a small multiple of the packet size. In presence of large buffers, this component becomes negligible.

7.1 Buffer sizes in practice

Remember that the FIFO size involves every queue in front of the actual serializer (the unit that takes packets and formats them into bits to be sent to the communication link). On most systems there are in fact at least two queues involved:

- a hardware FIFO, which is internal to the NIC and serves to absorb the latency in accessing system buses and memories;
- a software FIFO, typically called "transmit ring". This is a circular list of packet buffers used as a mailbox between the operating system and the NIC to queue packets to transmit. The role of this queue is to absorb the delay between the transmit completion interrupts and the software actually reacting and providing more packets to send.

considering that $W^k(t_1,t_2) \geq 0$, but the gain of at most $\Delta W/\Phi^k$ would not change the $1/\Phi^k$ behaviour of the RFI at the price of an unreadable expression.

The two queues serve two very different purposes and operate on largely different timescales. The hardware FIFO needs to amortize delays in the order of a few microseconds at most, which may occur when there is heavy contention on the system bus, e.g. because of long transactions.

Because the NIC cannot stop an ongoing transmission, the internal FIFO is normally storing at least one maximum-sized frame (1518 bytes on a standard ethernet, up to 10-16 Kbytes if jumbo frames are used). At 10..40 Gbit/s speeds, even a single frame might be too short to absorb bus delays, which is a reason why the internal FIFO is often much larger, between 32 and 512 Kbytes. The other reason for having large internal FIFOs is that many modern NICs implement TCP segmentation in hardware, so they must be able to store large TCP segments (typically up to 64 Kbytes each) to split them into separate IP packets.

The role of the transmit ring is instead to cope with possibly slow reaction from the operating system. With packet rates in the range of millions of packets per second, we cannot possibly expect the operating system to handle one interrupt per packet. Most modern NICs and operating systems implement a feature called interrupt coalescing/mitigation, where the maximum interrupt frequency is bounded, and latencies in the 20..100 μs range are quite common. On top of this, interrupt handlers might be delayed due to the CPU being busy on other processes, and 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.

In summary, practical values of ΔW on existing systems may easily be as large as $100L$ and more.

8. CONCLUSIONS

In this paper we have proved analytically the service properties of a large family of weighted fair queueing schedulers implementing the WF^{2}Q+ algorithm or fast, approximated versions, in presence of output FIFOs between the scheduler and the actual link, and without exact knowledge of the transmit rate.

The result is of practical relevance because such elements exist in all hardware, and it is important to know their impact when designing system with tight service guarantees.

In retrospect, our main results, summarized in Section 6, are perhaps obvious (but now we have a formal

Scheduler (ideal)	ΔS^k	ΔF^k	**B-WFI**	**RFI**
WF^{2}Q+	0	0	$L^k + 2\phi^k L$	–
S-KPS	$4\frac{L^k}{\phi^k}$	$2\frac{L^k}{\phi^k}$	$2\phi_k(\sigma_i + L)$	$2L + L^k/\phi_k + L^p/\phi_P + 3(\sigma_i + \sigma_j)$
GFQ	$\frac{L^k}{\phi^k}$	$\frac{L^k}{\phi^k}$	$3\phi^k(\sigma_i + 2L)$	$2L + L^k/\phi_k + L^p/\phi_P + 2(\sigma_i + \sigma_j)$
QFQ	$2\frac{L^k}{\phi^k}$	$4\frac{L^k}{\phi^k}$	$3\phi^k(\sigma_i + 2L)$	$3L + 4\sigma_i + 3\sigma_j$
FRR	n.a.	n.a.	not available	$4\sigma_i + 10\sigma_j + L/\phi^k$
Scheduler (real)	ΔS^k	ΔF^k	**B-WFI**	**RFI (simplified)**
AFQ			$2\phi^k \Delta W + \phi^k(\Delta S^k + \Delta F^k) + 2L^k - \phi^k L^k$	$2\max_k \left\{\Delta S^k + \Delta F^k + \Delta W + \frac{3L^k}{2\phi^k} - L^k\right\}$
WF^{2}Q+	0	0	$2\phi^k \Delta W + 2L^k - \phi^k L^k$	$2\max_k \left\{\Delta W + \frac{3L^k}{2\phi^k} - L^k\right\}$
S-KPS	$4\frac{L^k}{\phi^k}$	$2\frac{L^k}{\phi^k}$	$2\phi^k \Delta W + 8L^k - \phi^k L^k$	$2\max_k \left\{\Delta W + \frac{15L^k}{2\phi^k} - L^k\right\}$
GFQ	$\frac{L^k}{\phi^k}$	$\frac{L^k}{\phi^k}$	$2\phi^k \Delta W + 4L^k - \phi^k L^k$	$2\max_k \left\{\Delta W + \frac{7L^k}{2\phi^k} - L^k\right\}$
QFQ	$2\frac{L^k}{\phi^k}$	$4\frac{L^k}{\phi^k}$	$2\phi^k \Delta W + 8L^k - \phi^k L^k$	$2\max_k \left\{\Delta W + \frac{15L^k}{2\phi^k} - L^k\right\}$

Table 2: Service guarantees for various schedulers with ideal (no FIFO, exact $W(t)$) and real links (FIFO, $\overline{W}(t)$). For many of these algorithms, bounds are computed in terms of a parameter, $\sigma_k \approx L^k/\phi^k$ which determines how flows are grouped. We retain the original formulation for ease of reference. For the RFI we indicate a simplified expression assuming that the flows h and k that maximize the expression have the same parameters.

proof): the presence of a queue between the scheduler and the link affects the various service metrics by an amount roughly proportional to twice the queue size. The impact is less relevant for low-weight flows, which only use a small fraction of the total link capacity. More importantly, the scheduling perturbations induced by the queue may render completely irrelevant the difference in service properties among different schedulers.

9. REFERENCES

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

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

[3] Carbone, M., and Rizzo, L. Dummynet revisited. *ACM SIGCOMM Computer Communication Review 40*, 2 (2010), 12–20.

[4] Checconi, F., Valente, P., and Rizzo, L. QFQ: Efficient Packet Scheduling with Tight Bandwidth Distribution Guarantees. *http://info.iet.unipi.it/~luigi/qfq/*.

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

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

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

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

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

[10] Stiliadis, D., and Varma, A. A general methodology for designing efficient traffic scheduling and shaping algorithms. *Proceedings of IEEE INFOCOM '97* (April 1997), 326–335.

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

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

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