Analysis and Simulation of WF^{2}Q+ Based Schedulers: Comparisons and Compliance with Theoretical Bounds

Nicola Ciulli[1] and Stefano Giordano[2]

[1] Consorzio Pisa Ricerche - META Center
corso Italia, 116 - 56125 Pisa, Italy
n.ciulli@cpr.it
http://www.meta.cpr.it
[2] Dept. of Information Engineering - University of Pisa
Via Diotisalvi 2 - 56126 Pisa, Italy
giordano@iet.unipi.it
http://wwwtlc.iet.unipi.it

Abstract. The statistical multiplexing of non-fixed-size packet flows with heterogeneous requirements onto a single network interface of a router gave rise to a number of different scheduling mechanisms. These algorithms attempts to work as close as possible to the ideal *"fluid model"*. One of the most effective proposals is the *Worst-case Fair Weighted Fair Queueing (WF^{2}Q)*; this pays its optimality with a great computational complexity and has been followed by more "operative" derivatives: the WF^{2}Q+ and S-SPFQ. The former has not been specified with a univocal algorithm, thus leaving space for a number of implementations: this article aims to analyse and compare the different algorithms coming out from the WF^{2}Q concept at both theoretical and "practical" level (by means of simulations). To this purpose, an on-purpose discrete-event simulator for packet schedulers has been implemented at the University of Pisa. The simulations also allowed to verify to what extent the operative versions of WF^{2}Q fulfill to the WF^{2}Q theoretical properties.

1 Introduction

Forwarding a number of packet flows (i.e. inter-related sequences of packets) through a single network interface in a router requires the use of proper statistical multiplexing disciplines (i.e. packet schedulers) to regulate the access of packets from multiple queues to the interface. The purpose is to provide different shares of the service resource (i.e. bandwidth) to the different flows and, for this reason, they can be considered the basic functions to implement a service differentiation over IP networks. Other queueing disciplines (such as discarding functions) may be used, and have been implemented in our IP-QoS trial, but for sake of conciseness the tests shown in this article focus only on the scheduling aspect.

M. Ajmone Marsan, A. Bianco (Eds.): QoS-IP 2001, LNCS 1989, pp. 255–271, 2001.

A packet scheduling key problem rises from the basic fact that only one packet at a time can be transmitted through a network interface. This leads to consider as "merely ideal" the scheduler model where all the flows are served simultaneously (possibly at different "speeds" - i.e. shares of bandwidth). These models are often referred to as *Generalised Processor Sharing (GPS)* or *Fluid Fair Queueing (FFQ)* schedulers.

The problem of sharing link bandwidth among different flows (mapped into scheduler "connections") causes a number of alternative proposals aiming to realise the scheduling algorithm which better approximates the ideal behaviour of the GPS model.

1.1 The Problem of GPS Approximation

The "distance" of the approximation from its ideal model is measured through a set of properties and parameters, among which one of the most significant is the *"fairness"*, introduced later on. Here are listed some aspects under which a "real" packet scheduler may differ from the "ideal" one:

- since only one packet can be served each time, in different periods the whole link bandwidth is given to a single connection, and the link capacity is shared among the connections only in terms of their average values computed over "long" periods of time. (i.e. "long enough" to smooth such granularity in bandwidth allocation).
- In the GPS model, a packet entering a queue will wait for service for the time needed by the scheduler to *clear* the queue. In the real scheduler, the packet may experience a longer delay; e.g. if the packet enters an empty queue and the scheduler is busy, its transmission will not start immediately (as in the GPS model), but when the packet under service is finished.
- On the other hand, a packet may finish its service in the real scheduler sooner than in the GPS; e.g. if a packet enters an empty queue when the server is *idle*, its service time is L/C in the real scheduler (where L is the packet's size and C the link capacity), and L/r in the GPS (where r is the connection's service rate).
- As a consequence of the last two points, the connection queue occupancy in the real scheduler is different from the one in the GPS. The discrepancy between two corresponding queues should be limited and, possibly, small (the same applies to queueing delays).

1.2 Packet Scheduler "Fairness"

A *fair* scheduler manages the link bandwidth in such a way to give each connection an amount of service (i.e. time spent serving) proportional to its reserved rate, on *any* time interval. An *unfair* scheduler may offer a class, in some short periods, a service rate different from the allocated one. In real schedulers the shortest interval considered is the packet time, since during a packet service the whole link is owned by a single connection. The ideal behaviour is reached when

$w_i(t_1, t_2) = w_k(t_1, t_2) \quad \forall i, k \in B(t_1, t_2)$ (being $B(t_1, t_2)$ the set of backlogged connections and $w_i(t_1, t_2)$ is the service offered to class i, normalised to its allocated rate, in the interval $]t_1, t_2[$). The condition above is satisfied for any t_1, t_2 only by the GPS, whereas in a real system (since the granularity is the packet) it is never true at all and some indexes have been proposed to measure the scheduler behaviour in terms of fairness.

A good fairness index is the distance among the normalised services of the various connections on different time intervals. Another index is the *Worst-case Fair Index (WFI)* [1], which we focussed on; being r_i the service rate for class i, the WFI for scheduler S is defined as the value $C_{i,S}$ such that, for each packet, the following condition is true:

$$d_i^k \leq a_i^k + Q_{i,S}(a_i^k)/r_i + C_{i,S} \ , \tag{1}$$

where a_i^k is the arrival time of the packet at the i^{th} connection's queue, d_i^k is its departure time and $Q_{i,S}(t)$ is the queue occupancy at time t^+ (thus, $Q_{i,S}(a_i^k)$ includes the newly arrived packet).

1.3 The WF^{2}Q Algorithm

The WF^{2}Q has been introduced as a fairer variation of the basic WFQ structure. The WFQ [2][3] belongs to the class of *sorted-priority* schedulers [1] and is based on the emulation of the *corresponding* [2] GPS system, by means of a *"System Potential"* (or *"Virtual Time"*) function (which measures the overall work carried out by the GPS). When the WFQ is ready for service it chooses, among all the head-of-line packets, the one finishing its service first in the GPS, if no other packets were to arrive after the start of the service (this is a *SFF policy: Smallest virtual Finish time First*). The WFQ allows a good packet interleaving, but it may start serving also packets which would not have gone under service in the GPS yet, resulting in "bursts" of service on a single connection (for some reservation layouts and traffic inputs the WFQ's WFI increases linearly with the number of connections).

In order to overcome this weak point, the WF^{2}Q's set of service eligible packets is reduced to those packets which would have already started their service in the GPS (in the WFQ it is the whole group of head-of-line packets). This is an *SEFF (Smallest Eligible virtual Finish time First)* policy, and allows a very fine-grain interleaving of packets from the different connections and, consequently, a greater fairness.

[1] The *sorted-priority* schedulers carry out a packet-by-packet scheduling according to some time-varying priority assigned to each connection and provides a better fairness with respect to the *frame-based* schedulers (which divide the time into frames, assign a time-slot to each connection and serve bursts of packets in it – a famous example of them is the *Weighted Round Robin (WRR)*).

[2] Two packet schedulers are *corresponding systems* if they have the same configuration (same set of connections - in terms of queue length, service rate), and same incoming traffic patterns.

In the remainder of this article, section 2 introduces some details on the WF^{2}Q-based scheduling algorithms and outlines the symbolism which will be used throughout the article. In section 3 some comparisons among packet schedulers are carried out.

2 Overview of WF^{2}Q-Based Scheduling Algorithms

2.1 Some History

As introduced in section 1.3, the main computational burden in the WFQ and WF^{2}Q implementations is the calculation of a system potential function. Thus, the WF^{2}Q proposal has been followed by many "operative" algorithms which try to approximate it. Among these schedulers, generically named as "WF^{2}Q+", some descend directly from the original WF^{2}Q+ definition [4], whereas one (the *Shaped Starting Potential Fair Queueing – S-SPFQ*) is the outcome of an independent work [5] (placed under the "WF^{2}Q+" class since it is actually an operative version of the WF^{2}Q). The following list provides a bit of information on these WF^{2}Q+ schedulers, which will be analysed more thoroughly later on:

- The S-SPFQ is the only well-defined packet scheduler, in the sense that a real algorithm is provided with it: the system potential approximation comes along with an algorithm which tells when the potential update should be done (this piece of information is needed, as explained in section 2.3).
- The *"Varma" WF^{2}Q+ (V-WF^{2}Q+)* is a WF^{2}Q+ inspired by the S-SPFQ algorithm structure, but with a different set of assumptions; the V-WF^{2}Q+ is defined in this article in order to explore one more direction for the WF^{2}Q operative schedulers, mainly for comparison purposes.
- The *"Zhang" WF^{2}Q+ (Z-WF^{2}Q+)* has been "hinted" at in [4], i.e. some information is missing to get an algorithm out of it (see section 3.2. The Z-WF^{2}Q+ (which should, in Zhang's honour, be referred to as *the* WF^{2}Q+) algorithm can be inferred from some considerations in literature and from a practical implementation of it (the *C-WF^{2}Q+*, a piece of C++ code for the NS 2.1 simulator [8]). But the C-WF^{2}Q+ algorithm seems to be substantially different from Z-WF^{2}Q+. See section 3.2 for further details.

2.2 Approximated System Potential Formulas for the WF^{2}Q+

In order to overcome the calculation of the Virtual Time function, which is a very heavy duty in a real-time implementation, Zhang proposed [4] a Virtual Time function approximation, which is recursively updated on the basis of the overall service amount carried out by the *real scheduler* [3]:

$$V_S(t_2) = max\left\{V_S(t_1) + W(t_1, t_2),\ \min_{i \in B(t_2)} SP_i^{h_i(t_2)}\right\}, \tag{2}$$

[3] Another approximated Virtual Time function appears in [5] (where the S-SPFQ is defined), with a different notation: "Potential" (P) instead of "Virtual Time" (V), "Starting, Finishing Potential" (SP, FP) instead of "Starting, Finishing Time" (S, F). In this article these basically equivalent concepts will be used indifferently.

where $W(t_1, t_2)$ is the amount of work carried out by the scheduler S in $[t_1, t_2]$. If the scheduler is constantly active in such a period, $W(t_1, t_2) = t_2 - t_1$. $B(t_2)$ is the set of backlogged connections at t_2 and $h_i(t_2)$ is the index of the head-of-line packet of connection i at time t_2: i.e. $min_{i \in B(t_2)} SP_i^{h_i(t_2)}$ is the smallest starting potential at time t_2. As can be noticed, the potential at a given time is obtained from the potential at a previous time and from the work carried out by the real scheduler in between.

Another layout for the above formula exists [7], and might seem different:

$$\widetilde{V}_S(t_2) = max \left\{ \widetilde{V}_S(t_1) + W(t_1, t_2), \min_{i \in B(t_1)} SP_i^{h_i(t_1)} \right\} . \tag{3}$$

The difference is that the minimum SP is evaluated at time t_1 instead of t_2. The two formulas can be considered the same if $t_1, t_2 \in]t_s, t_e]$ (a single "service interval"), where t_s is the starting time of the current service and t_e is the finish time of the current service. In fact, the minimum SP does not change in $]t_s, t_e]$, as explained in the proof of theorem 2.

2.3 "Transitivity" of Zhang's WF^{2}Q+ System Potential Formula

Zhang's formula allows to evaluate the potential at time t by knowing the potential at a previous time t_0, the work done by the scheduler in $]t_0, t[$ and the minimum SP among all the backlogged queues at time t (briefly indicated as $SP_{min}(t)$). Thus, the $V(t)$ function can be written as:

$$V(t) = f\left(V(t_0), W(t_0, t), SP_{min}(t)\right) . \tag{4}$$

Theorem 1. *With the above potential definition, given t_0, t_1, t_2 belonging to the same service interval, the potential value at time t_2 obtained from the status at time t_0 is different from the potential calculated starting from the status at time t_1, being the potential at time t_1 obtained by the status at time t_0. That is, defining*

$$V(t_2) = f\left(V(t_0), W(t_0, t_2), SP_{min}(t_2)\right) , \tag{5}$$

$$\widehat{V}(t_2) = f\left(V(t_1), W(t_1, t_2), SP_{min}(t_2)\right) , \tag{6}$$

being $V(t_1) = f(V(t_0), W(t_0, t_1), SP_{min}(t_1))$,

$$\exists\, t_0, t_1, t_2 \in]t_s, t_e] \quad | \quad V(t_2) \neq \widehat{V}(t_2) . \tag{7}$$

Proof. Let's write explicitly the formulas:

$$V(t_2) = max\left\{V(t_0) + W(t_0, t_2), SP_{min}(t_2)\right\} ; \tag{8}$$

$$\widehat{V}(t_2) = max\left\{V(t_1) + W(t_1,t_2), SP_{min}(t_2)\right\} = \\ = max\left\{max\left\{V(t_0) + W(t_0,t_1), SP_{min}(t_1)\right\} + W(t_1,t_2), SP_{min}(t_2)\right\} \ . \quad (9)$$

$SP_{min}(t)$ is constant $\forall t \in]t_s, t_e]$, since the set of head-of-line packets at t_s does not change and any newly arrived packet on an empty queue (which causes a pool change) will be assigned an SP greater than the $min(SP)$. Let's call K this constant value: $SP_{min}(t_1) = SP_{min}(t_2) = K$; thus,

$$\widehat{V}(t_2) = max\left\{max\left\{V(t_0) + W(t_0,t_1), K\right\} + W(t_1,t_2), K\right\} = \\ = max\left\{V(t_0) + W(t_0,t_1), K\right\} + W(t_1,t_2) \ . \quad (10)$$

Now, there can be two sub-cases:

1. if $V(t_0) + W(t_0,t_1) \geq K$:

$$\widehat{V}(t_2) = V(t_0) + W(t_0,t_1) + W(t_1,t_2) = V(t_0) + W(t_0,t_2) \quad (11)$$

(note that $\forall t_a, t_b, t_c \ \ W(t_a,t_b) + W(t_b,t_c) = W(t_a,t_c)$). Then, since $V(t_0) + W(t_0,t_2) \geq K$ (being $W(t_0,t)$ a monotonous non decreasing function in t), we have:

$$V(t_2) = V(t_0) + W(t_0,t_2) \quad (12)$$

thus, in this case, $V(t_2) = \widehat{V}(t_2)$.

2. but, if $V(t_0) + W(t_0,t_1) < K$, the potential values are:

$$\widehat{V}(t_2) = K + W(t_1,t_2) \quad (13)$$

$$V(t_2) = \begin{cases} \text{if } V(t_0) + W(t_0,t_2) \geq K & V(t_0) + W(t_0,t_2) \\ \text{if } V(t_0) + W(t_0,t_2) < K & K \end{cases} < \widehat{V}(t_2) \quad (14)$$

both values are possible for $V(t_2)$, depending on $W(t_1,t_2)$; for both $V(t_2)$ values, $V(t_2) < \widehat{V}(t_2)$.

□

The above proof shows that the potential updating formula is not enough to define a WF^{2}Q+ scheduler, and the $V(t)$ formula should be re-defined as $V(t,t_0)$. The specification of a potential updating algorithm is also needed. Two scheduling algorithms which use such an approximated potential function and update it at different times may be different, but it is not evident that the different potentials induce different output packets ordering. This aspect has been investigated by means of simulations and presented in section 3.

2.4 Overview of the Algorithms

The symbols used are the following:

1. p_i^k is the k^{th} packet of connection i, L_i^k its length and a_i^k its arrival time;
2. SP_i^k and FP_i^k are, respectively, the starting and finishing potentials of p_i^k;
3. t_e is the time at which the last service ended;
4. $V(t_e^+)$ is the potential evaluated at t_e, but after the calculations done at the end-of-service.

A note on the Z-WF^{2}Q+ and C-WF^{2}Q+: the two algorithms differ from the S-SPFQ and V-WF^{2}Q+ models in that the last ones use the SP and FP variables for each packet in the queue system, whereas the first ones use one (SP, FP) couple for the whole connection, thus reducing the complexity.

Furthermore, S-SPFQ and V-WF^{2}Q+ take actions at three events: *(1)* when enqueueing a packet, *(2)* when deciding on the next packet to send and *(3)* when ending the service of a packet. Z-WF^{2}Q+ and C-WF^{2}Q+, instead, are based on an action to be taken when a packet reaches the head of its queue and on an action taken when scheduling the next packet. Anyway, their "two-triggering-events" scheme can be easily mapped into S-SPFQ's and V-WF^{2}Q+'s "three-triggering-events" scheme (on which we based our table), by distinguishing the two possible times when a packet can reach the head of its queue: when being enqueued on an empty queue or when the packet ahead of it is scheduled for service.

The different scheduling algorithms are summarised in table in Figure 1.

3 Commonalities and Differences

A thorough set of comparisons between the WF^{2}Q derivatives is here presented. The comparisons are carried out at theoretical level and with the support of simulations, as well. The simulation tool is a discrete-event simulator ("*sch_sim*") tailored for packet scheduling algorithms, implemented at the Information Engineering Dept. NetLab of University of Pisa. *Sch_sim* processes input traffic traces running a number of scheduling algorithms along with the corresponding GPS system. Two relevant features are the exact implementation of WFQ and WF^{2}Q algorithms and assessment of theoretical properties of an algorithm (e.g. differences between the real scheduler and GPS queues lengths).

3.1 S-SPFQ and V-WF^{2}Q+

The two algorithms share the same "packet selection" and "end-of-service" operations. The difference is in the "packet arrival" operations, where they have two different ways to calculate the SP of the newly arrived packet. Now, the two algorithms use different terms in the second member of the max expression. The V-WF^{2}Q+ uses

$$V_{VWF^2Q+}(a_i^k) = max\left\{V_{VWF^2Q+}(t_s) + a_i^k - t_s,\ \min_{i\in B(a_i^{k-})} SP_i^{h_i(a_i^{k-})}\right\}, \quad (15)$$

	Packet arrival	Packet selection	End-of-service
S-SPFQ	❑ If the scheduler is serving a packet: $SP_i^k = \max\{FP_i^{k-1}, V(t_S) + a_i^k - t_S\}$ $FP_i^k = SP_i^k + L_i^k / r_i$ ❑ If the scheduler is idle: *reset all the parameters.*	*Choose the packet with lowest FP, among all the head-of-line packets with* $SP \leq V(t_E+)$.	❑ Potential update: $V(t_E) = \max\left\{V(t_S) + L_i^k / r_i, \min_{i \in B(t_E^-)} SP_i^{h_i(t_E^-)}\right\}$
V-WF²Q+	❑ If the scheduler is serving a packet: $SP_i^k = \max\{FP_i^{k-1}, V(a_i^k)\}$ $FP_i^k = SP_i^k + L_i^k / r_i$ ❑ If the scheduler is idle: *reset all the parameters.*	*Choose the packet with lowest FP, among all the head-of-line packets with* $SP \leq V(t_E+)$.	❑ Potential update: $V(t_E) = \max\left\{V(t_S) + L_i^k / r_i, \min_{i \in B(t_E^-)} SP_i^{h_i(t_E^-)}\right\}$
Z-WF²Q+	❑ If the packet finds an empty queue: $SP_i \leftarrow \max\{FP_i, V(a_i^k)\}$ $FP_i \leftarrow SP_i + L_i^k / r_i$	❑ *Choose the head-of-line packet of the connection with lowest FP, among all the connections with* $SP \leq V$. ❑ Then: $SP_i \leftarrow FP_i$ $FP_i \leftarrow SP_i + L_i^k / r_i$	
C-WF²Q+	❑ If the packet finds an empty queue: $SP_i \leftarrow \max\{FP_i, V\}$ $FP_i \leftarrow SP_i + L_i^k / r_i$ $V \leftarrow \max\{V, SP_{MIN}(a_i^{k-})\}$	❑ *Choose the head-of-line packet (* p_i^h *) of the connection with lowest FP, among all the connections with* $SP \leq V$. ❑ If there's a packet p_i^{h+1} after dequeueing p_i^h: $SP_i \leftarrow FP_i$ $FP_i \leftarrow SP_i + L_i^{h+1} / r_i$ ❑ In any case, update: $V \leftarrow \max\{V + L_i^h / r, SP_{MIN}(t_S^+)\}$	

Fig. 1. Example of S-SPFQ and V-WF^{2}Q+ system potential evolution

where t_s is the current service start time. Note that SP_{min} is calculated at a_i^{k-}: the newly arrived packet is not taken into account when evaluating the minimum SP, being its SP not yet defined. This is quite reasonable, thus the "-" sign in a_i^{k-} will not be used in the remainder of this document, because $SP_{min}(a_i^{k-})$ will unambiguously indicate the minimum SP evaluated at the packet arrival on a pool of head-of-line packets which does not include the newly arrived packet. It's easy to see that the S-SPFQ formula for the SP calculation is a simplification of the V-WF^{2}Q+'s one; in fact, S-SPFQ uses

$$V_{SSPFQ}(a_i^k) = V_{SSPFQ}(t_s) + a_i^k - t_s \ . \tag{16}$$

Explanation of the S-SPFQ Potential Updating Formula at Packet Arrival. We can set out the problem according to the following theorem.

Theorem 2. *The S-SPFQ and V-WF^{2}Q+ use the same potential formula (15), but, once started a packet's service, the V-WF^{2}Q+ excludes the packet's SP from the pool $B(t)$ over which $min(SP)$ is evaluated, whereas the S-SPFQ keeps its SP in that pool, and extract it just before ending its service and calculating the new potential. Thus, the $min(SP)$ calculated at end-of-service is the same for both schedulers ($\Rightarrow$ the potentials are the same at that time), whereas the ones evaluated at packet arrival times (during a service) are different ($\Rightarrow$ the potentials may be different).*

Proof. Here it will be proved that the S-SPFQ's different management of the served packet's SP leads to the simplified formula (16).

Be $]t_s, t_e]$ the service interval considered; since the packet under service is not included in the V-WF^{2}Q+'s set of packets for the $min(SP)$ evaluation, $SP_{min}(t)$ is constant in $]t_s, t_e]$ [4] and has a point of discontinuity in t_s: $SP_{min}(t_s^-) \neq SP_{min}(t_s^+)$. The $B_{SSPFQ}(t)$ set, on the contrary, includes the packet under service, until t_e^-, that is just a "thick" before its end-of-service time; then, the SSPFQ $SP_{min}(t)$ is constant in $[t_s, t_e[$ and has a point of discontinuity in t_e: $SP_{min}(t_e^-) \leq SP_{min}(t_e^+)$. Of course, a t_e is the "t_s" of the next interval. It is interesting that, for the S-SPFQ, the $min(SP)$ evaluated at the end of the previous service is the same as evaluated at packets arrivals in $[t_s, t_e[$. Let's refer to the $min(SP)$ value in $[t_s, t_e[$ as K:

$$V_{SSPFQ}(t_s) = V_{SSPFQ}(t_{e_{prev}}) = \\ = max\left\{V_{SSPFQ}(t_{s_{prev}}) + t_{e_{prev}} - t_{s_{prev}}, K\right\} \geq K \ , \tag{17}$$

where $t_{s_{prev}}$ and $t_{e_{prev}}$ are, respectively, the starting and ending times of the previous service. Note that $t_s = t_{e_{prev}}$, and is the instant soon before the starting of the current service, when the potential is updated (being the set of SPs updated), as explained in the following. Thus, a-fortiori

$$\forall a_i^k \in [t_s, t_e[\quad V_{SSPFQ}(t_s) + a_i^k - t_s \geq SP_{min}(t_s) = SP_{min}(t_s^+) = K \ . \tag{18}$$

□

Explanation of the Notation. A clarification is needed on the notation used to express the end-of-service (≡ beginning-of-service) times (t_{s,e^-} , $t_{s,e}$, t_{s,e^+}). At these times, both S-SPFQ and V-WF^{2}Q+ perform instantaneously two main changes to the system status: *(1)* the potential function is updated and *(2)* a new packet is scheduled for transmission – and the $min(SP)$ either changes (V-WF^{2}Q+) or not (S-SPFQ). In order to distinguish among the various steps at the same instant $t_{s,e}$, the following notation is used in this article:

- $t_{s,e}^-$ is the time right before the potential update; nothing has changed yet.
- $t_{s,e}$ indicates that the potential has been updated (for the S-SPFQ, this means that the ended packet's SP has been removed from the pool), but a new packet has not been scheduled yet.
- $t_{s,e}^+$ indicates that the new packet has been scheduled; in case of V-WF^{2}Q+, the packet's SP has been removed from the pool over which $min(SP)$ is calculated; everything now has been done.

Thus, $V(t_{s,e}^-)$ is the potential right before the update, $V(t_{s,e})$ is the updated potential and (in the V-WF^{2}Q+ case) $V(t_{s,e}^+)$ is a new potential value, since it is calculated as the maximum between $V(t_{s,e})$ and the (possibly) new $min(SP)$.

[4] For both V-WF^{2}Q+ and S-SPFQ, a new packet arrival on an empty queue in $]t_s, t_e]$ may change the $B(t)$ set but not the $min(SP)$, since the new packet's $SP \geq V(t_s) \geq SP_{min}(t_s)$, and $SP_{min}(t_s)$ belongs to a packet in the system in $]t_s, t_e]$.

Now we can better explain the statement on the $min(SP)$ discontinuities. In both V-WF^{2}Q+ and S-SPFQ, the $min(SP)$ has a discontinuity at the end (beginning) of services. But in the V-WF^{2}Q+ the discontinuity is associated to action 2, i.e. between $t_{s,e}$ and $t^+_{s,e}$, whereas in the S-SPFQ the discontinuity is associated to action 1, i.e. between $t^-_{s,e}$ and $t_{s,e}$.

Temporal Evolution of V-WF^{2}Q+'s and S-SPFQ's System Potential. This section provides a graphical representation example of the qualitative trends of S-SPFQ and V-WF^{2}Q+ system potentials. The right plot in Figure 2 allows to see the two potential overlaid. The parts where $V_{SSPFQ}(t)$ differs from $V_{VWF^2Q+}(t)$ are represented as dashed lines.

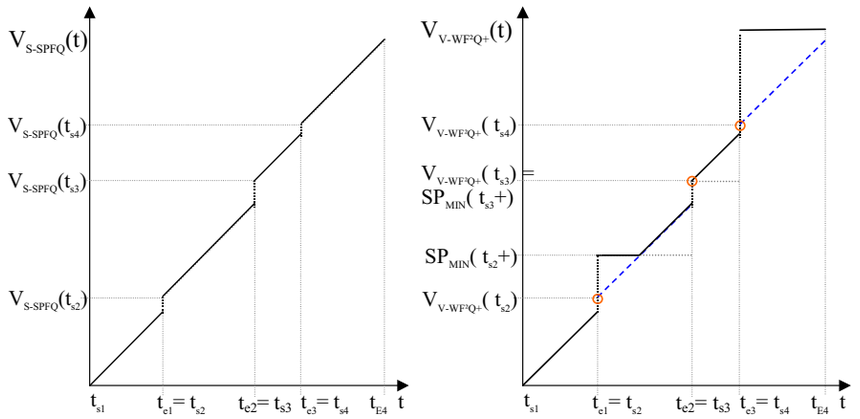

Fig. 2. Example of S-SPFQ and V-WF^{2}Q+ system potential evolution

Difference between V-WF^{2}Q+ and S-SPFQ SPs Evaluated at Packet Arrivals. The different S-SPQF and V-WF^{2}Q+ potential formulas may cause different potential values to be evaluated at a packet arrival (assumed that the server is busy) and, consequently, a different Starting Potential assigned to that packet. A further discussion is needed on the last consequence. First, let's recall that ${}^{S}SP_i^k = max\left\{{}^{S}FP_i^{k-1}, V_S(a_i^k)\right\}$, where "$S$" is either S-SPFQ or V-WF^{2}Q+. Using the formulas (15) and (16), the SPs can be written as:

$$ {}^{VWF^2Q+}SP_i^k = max\left\{{}^{VWF^2Q+}FP_i^{k-1}, V_{VWF^2Q+}(t_s) + a_i^k - t_s, SP_{min}(a_i^k)\right\} \tag{19}$$

$$ {}^{SSPFQ}SP_i^k = max\left\{{}^{SSPFQ}FP_i^{k-1}, V_{SSPFQ}(t_s) + a_i^k - t_s\right\} \tag{20}$$

Let's divide our discussion into two cases:

1. If packet arrives at a_i^k and finds an empty queue, and its previous packets was served "enough" time before a_i^k, it may be true that

$$FP_i^{k-1} \ \left(= SP_i^{k-1} + L_i^{k-1}/r_i\right) < V_S(t_s^+) \ \left(\leq V_{SSPFQ}(a_i^k) \leq V_{VWF^2Q+}(a_i^k)\right) \tag{21}$$

   this is true if SP_i^{k-1} is lower "enough" than the $min(SP)$ which contributes to $V_S(t_s^+)$ (i.e. if p_i^{k-1} was served enough time ago). Here we used the property that the SP of a packet is always higher than the SP of a previously arrived packet. If, in addition, $SP_{min}(a_i^k) \leq V_{VWF^2Q+}(t_s) + a_i^k - t_s$, we get:

$$^{VWF^2Q+}SP_i^k =^{SSPFQ} SP_i^k = V_S(t_s) + a_i^k - t_s \tag{22}$$

   (remember that V-WF^{2}Q+'s and S-SPFQ's potentials have the same value right after the beginning of a service). This, actually, proves that the SPs assigned to a packet by V-WF^{2}Q+ and S-SPFQ may be equal.
2. But, in any other case, e.g. when a packet enters a backlogged session, the above assumptions can't be made, and the new SP may be equal either to the previous packet's FP (which may be different for the two schedulers) or to system potential (which may be different for the two schedulers).

Are the V-WF^{2}Q+ and S-SPFQ Different? The possibility that different SPs are assigned by the V-WF^{2}Q+ and S-SPFQ to a newly arrived packet, may not prevent the two schedulers from having always the same output packets sequence (thus being equivalent systems).

The difference between the V-WF^{2}Q+ and the S-SPFQ systems can be proved with just a single example where the two schedulers produce different output sequences under the same input sequence; thus, the proof has been found using the *sch_sim* simulator, configured with different scenarios. In some cases, the two schedulers actually resulted in different outputs. Here in the following, one of these situations is described.

- General configuration:

```
input from '2_video_flows', output into 'wf2qplus.outseq.1',
duration= 60 sec, link rate = 250 Kb/s, 2 queues:
Q[1]:    maxlength = 100000 B,  service rate = 170 Kb/s
Q[2]:    maxlength = 100000 B,  service rate = 80 Kb/s
```

- The output of the two systems has been exactly the same until $t = 308.91$ ms. At this time the situation is the following:
  - connection 1 has a queue level = 0 bytes;
  - connection 2 has a queue level = 3331 bytes; the head-of-line packet has SP=314.606 ms and **FP=352.006 ms** and a packet under service (FP = 314.606 ms), started at $t = 297.293$ ms and finishing at $t = 309.261$ ms.

– then, at $t = 308.91$ ms, a new packet (1064 bytes) arrives at queue 1.
  - the V-WF^{2}Q+ assigns the following values:
    ```
    ENQUEUE: q[1]: t=308.910000, size=1064, qlen_found=0,
             pkt=0x58738, pot=277.206000(+T=288.823000),
             sp=314.606000, fp=364.676588 (prev=269.080353),
              min_sp=314.606000[2,0x58408], pot_used=314.60600}
    ```
  - the S-SPFQ assigns the following values:
    ```
    ENQUEUE: q[1]: t=308.910000, size=1064, qlen_found=0,
             pkt=0x58738, pot=277.206000, sp=288.823000,
             fp=338.893588 (prev=269.080353), pot_used=288.8230
    ```

– at $t = 309.261$ ms, the current service ends and a new packet is scheduled:
  - the V-WF^{2}Q+ updates the potential:
    ```
    EOS    : Q[2]: t=309.261000, pkt=0x584e0, gps_pkt=0x58c50
             pot=314.606000, min_sp=314.606000[1, 0x58788]
    ```
    and makes the following choice:
    ```
    SERVNXT: t=309.261000
             Q[2]: pkt=0x58458, size=374, min_fp=352.006000,
             pot=314.606000
    ```
  - the S-SPFQ updates the potential:
    ```
    EOS    : Q[2]: t=309.261000, pkt=0x584e0, gps_pkt=0x58c50
             pot=289.174000, min_sp=288.823000[1, 0x58788]
    ```
    and makes the following choice:
    ```
    SERVNXT: t=309.261000
             Q[1]: pkt=0x58788, size=1064, min_fp=338.893588,
             pot=289.174000
    ```

That is, in this case, the two schedulers choose different packets for transmission. *Thus, we proved that at least one case exists where V-WF^{2}Q+'s and S-SPFQ's different SPs (and FPs) cause a different scheduling decision.*

3.2 Z-WF^{2}Q+ and C-WF^{2}Q+

Some notes are required on the two algorithms before beginning a comparison:

– In the Z-WF^{2}Q+, the potential updating times are not specified; the potential used to calculate the SP of a packet which reaches the head of an empty queue should be evaluated at the arrival time of the packet (see Figure 1). But, as shown in section 2.2, the potential formula is not "transitive", and the potential value at a given time depends on the updating algorithm. Thus, saying "use V evaluated at the arrival time of the packet" is not sufficient.

- The C-WF^{2}Q+ updates V when enqueueing a packet on an empty queue. To this purpose, the minimum SP is calculated just before enqueueing the packet: this is the meaning of the "$-$" sign in a_i^{k-}.
- The C-WF^{2}Q+ updates V after scheduling a packet for service. In the calculation of the minimum SP at this time (t_s), the scheduled packet is considered already removed from its queue: this is the meaning of the "+" sign after t_s^+.

Discussion on the Z-WF^{2}Q+ and C-WF^{2}Q+. The comparison between Z-WF^{2}Q+ and C-WF^{2}Q+, is made difficult by the fact that *the Z-WF^{2}Q+ is not a completely specified algorithm* (in the sense reported above).

Here the C-WF^{2}Q+ is assumed to be the "official" implementation of the Z-WF^{2}Q+ scheduler, and its potential updating policy will be inspected. In fact, the C-WF^{2}Q+ appears to be using, at each event, the potential value updated at the end of the previous event, and this may result in a unclear evaluation of the potential used in the SP calculation.

As can be noticed, then, the potential updated at the beginning of a packet's service is anticipated with respect to its real value: in fact, it is immediately added with the packet's service time (i.e. Zhang's formula is not applied). Then, when a packet is enqueued, the potential is updated, but it is not added with the amount of service done by the scheduler since last event (i.e. Zhang's formula is not applied), probably due to the fact that the whole service done for the packet under transmission has already been taken into account.

A simple example is enough to show that the C-WF^{2}Q+ does not use the "correct" potential value when enqueueing a packet (on an empty queue):

- initially the whole system is idle (no packets): the potential is zero;
- at t_1 a packet arrives at connection i and is served;
- at t_2 a second packet arrives at connection j, with the previous packet still under service.
- Now, the real potential value should be $t_2 - t_1$ (and this also should be the corresponding GPS potential value), but the C-WF^{2}Q+ already increased V of the whole service time of the first packet at its beginning-of-service time: thus, V is higher in C-WF^{2}Q+ than it should.

Thus, this example shows that the C-WF^{2}Q+ potential may be far from the GPS potential.

Definition of a Z-WF^{2}Q+ Algorithm. As explained before, the potential value at a given time is defined by both an updating formula and an updating algorithm (which specifies when the potential value has to be updated). The comparison between the C-WF^{2}Q+ and the Z-WF^{2}Q+ needs some further assumptions on the latter in order to be completed.

In [6], Varma says that "In an idelized fluid server it is possible to update the system potential at any instant of time. However, in a packet-by-packet server it

is desirable to update the system potential only when a packet departs the system." This choice unambiguously defines an updating policy and, consequently, the time evolution of the system potential.

The following algorithm is the result of the above consideration applied to the Z-WF^{2}Q+:

- *packet arrival on an empty queue*: $SP_i \leftarrow max\{FP_i, V(a_i^k)\}$ and $FP_i \leftarrow SP_i + L_i^k/r_i$;
- *packet scheduling*: if a service just ended, update the potential [5] and choose the head-of-line packet of the lowest FP connection among all the connections with $SP \leq V$; then update: $SP_i \leftarrow FP_i$ and $FP_i \leftarrow SP_i + L_i^k/r_i$.

A set of simulations carried out on *sch_sim* showed that the C-WF^{2}Q+ and this Z-WF^{2}Q+ (with potential update at end-of-service) produce different output packet sequences (i.e. are different algorithms).

3.3 Z-WF^{2}Q+ and V-WF^{2}Q+ / S-SPFQ

The most evident difference between the V-WF^{2}Q +/S-SPFQ and Z-WF^{2}Q+ (as previously defined) approaches is the association of SPs and FPs with the single packet in the former cases, and with the whole connection in the latter. In this section we will show that such a difference leads to a different output.

The Z-WF^{2}Q+ algorithm says that the SP and FP of the connection have to be updated when a packet reaches the head of the queue:

- if the packet arrived at an empty queue: $SP_i \leftarrow max\{FP_i, V(a_i^k)\}$;
- if the packet found at least a packet ahead at its arrival: $SP_i \leftarrow FP_i$.

This is equivalent to update SP as soon as a packet arrives and to bind its SP with it:

- if the packet arrived at an empty queue: $SP_i^k = max\{FP_i^{k-1}, V(a_i^k)\}$;
- if the packet found at least a packet ahead at its arrival: $SP_i^k = FP_i^{k-1}$.

The "empty queue" case formula is the same as in the V-WF^{2}Q+ and S-SPFQ. The second case formula, instead, is based on the assumption that $FP_i^{k-1} \geq V(a_i^k)$, where $V(a_i^k) = max\{V(t_s) + a_i^k - t_s, SP_{min}(a_i^k)\}$. Now, since p_i^{k-1} was still in queue at a_i^k, it is true that $FP_i^{k-1} \geq SP_i^{k-1} \geq SP_{min}(a_i^k)$; but we cannot say that $FP_i^{k-1} \geq V(t_s) + a_i^k - t_s$.

Simulations showed that some cases exist where the Z-WF^{2}Q+ produces a different output with respect to either V-WF^{2}Q+ and S-SPFQ.

3.4 C-WF^{2}Q+ and V-WF^{2}Q+ / S-SPFQ

The same discussion applied previously when showing the difference between Z-WF^{2}Q+ and C-WF^{2}Q+ can be now used to explain the difference between the C-WF^{2}Q+ and both the V-WF^{2}Q+ and S-SPFQ. Simulation results confirm this.

[5] $V \leftarrow max\left\{V + L_i^{k-1}/r_i, min_{i \in B(t_e^-)} SP_i^{h_i(t_e^-)}\right\}$

3.5 Final Considerations on the Comparisons

The above comparisons showed that S-SPFQ, V-WF^{2}Q+, Z-WF^{2}Q+ and C-WF^{2}Q+ are all different algorithms, in the sense that they may produce different packet sequences under some circumstances. This may not prevent them from having equivalent performances, as is discussed in the following.

4 WF^{2}Q+ Schedulers' Compliance with WF^{2}Q Bounds

Another set of simulations showed that each implementation of a WF^{2}Q+ algorithm is different from the original WF^{2}Q (producing a different packet sequence). What is relevant, then, is to check if the WF^{2}Q+ implementations still fulfill the WF^{2}Q fairness property [6]. This property comes out from the following *sufficient* conditions characterising the WF^{2}Q:

$$\forall i,k \quad d^k_{i,WF^2Q} - d^k_{i,GPS} \leq L^{MAX}/r \ , \tag{23}$$

$$\forall \tau \quad Q_{i,GPS}(\tau) - Q_{i,WF^2Q}(\tau) \leq (1 - r_i/r)L_i^{MAX} \ , \tag{24}$$

and from the GPS property: $\forall i,k \ \ d^k_{i,GPS} - a^k_i \leq Q_{i,GPS}(a^k_i)/r_i$.

The simulations showed that for each WF^{2}Q+ implementation some cases occur where one of the first two (or both) conditions are *not* satisfied. Here is a sample of such simulations on the C-WF^{2}Q+; similar results apply to the other WF^{2}Q+ schedulers [7]:

```
sch_sim:simulation parameters:
        input from '2_video_flows', output into 'cwf2q+.outseq.1',
        duration= 60 sec
        scheduler = 'c-wf2q+'(40), link rate = 250 Kb/s, 2 queues:
        Q[1]:    maxlength = 100000 B,  service rate = 170 Kb/s
        Q[2]:    maxlength = 100000 B,  service rate = 80 Kb/s

Results:
Q[1]:   servc:  sent: 1223786 B, 2290 pkts; recv: 1223786 B,
                2290 pkts; drops: 0 pkts, avg_rate= 163.17 Kb/s
        delays: avg= 320.36 ms, max= 1117.21 ms,
                wfi= 45.783 ms (th:50.353 [ok]),
                Dsch-Dgps_max= 48.473 ms (th:34.240 [no])
        queues: max_real= 23634 B, max_gps= 23490.3 B,
                sched-gps_min= -799.4 B (th:-342.4 [no]),
                sched-gps_max= 1149.7 B (th:1070 [no])
```

[6] Each connection WFI should be lower than the WF^{2}Q bound defined in [1].

[7] For each listed connection i, `sched-gps_min` is $min\{Q_{i,WF^2Q}(\tau) - Q_{i,GPS}(\tau)\}$ for $\tau \in$ the simulation interval, and `Dsch-Dgps` is the $max_k\{d^k_{i,WF^2Q} - d^k_{i,GPS}\}$ for the connection. Each parameter is followed by its theoretical value ("`th`").

```
Q[2]:   servc:  sent: 651214 B, 1682 pkts; recv: 998258 B,
                2023 pkts; drops: 341 pkts, avg_rate= 86.82 Kb/s
        delays: avg= 7466.13 ms, max= 9981.71 ms,
                wfi= 58.648 ms (th:107.000 [ok]),
                Dsch-Dgps_max= 61.141 ms (th:34.240 [no])
        queues: max_real= 99990 B, max_gps= 99999.2 B,
                sched-gps_min= -1149.7 B (th:-727.6 [no]),
                sched-gps_max= 799.4 B (th:1070 [ok])
```

Obviously, this does not prevent the fairness theoretical bound from being fulfilled by the WF^{2}Q+ schedulers; in fact, all the simulations showed that such a bound is still true for the Z-,V-,C-WF^{2}Q+ and for the S-SPFQ, as well.

This leads to conclude that, notwithstanding the differences among the four WF^{2}Q+ algorithms, they still perform within the theoretical bound and can be used indifferently when a WF^{2}Q level of fairness is required.

5 Conclusions

This article summarises the work done around the "most fair" WF^{2}Q scheduler, discussing at both theoretical and simulation level the differences among its WF^{2}Q+ implementations. Some of them are already clearly defined (S-SPFQ and C-WF^{2}Q+), whereas others are "interpolated" from basic ideas and proposals in literature. At the end of the analysis process, each one of the WF^{2}Q+ implementation showed to be different from the others, since the produced output packet sequences differ in some cases. These differences, however, does not affect their performance equivalence (in terms of delay and fairness bounds).

Acknowledgements: This work was partially carried out with the support of the Italian Ministry of University and Scientific Research (MURST Contract N. 9809321920) "Techniques for quality of service guarantees in multiservice telecommunication networks".

References

1. J.C. Bennet, H. Zhang: WF^{2}Q: Worst-case Fair Weighted Fair Queueing. INFOCOM '96 (1996)
2. A. Demers, S. Keshav, S. Shenkar: Analysis and Simulation of a Fair Queueing Algorithm. Internet. Res. and Exper., vol. 1 (1990)
3. A.K. Parekh, R. Gallager: A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. IEEE/ACM Trans. on Networking, Vol. 1 , No. 3 (1993)
4. J.C. Bennet, H. Zhang: Hierarchical Packet Fair Queueing Algorithms. IEEE/ACM Trans. on Networking, Vol. 5, No. 5 (1997)
5. A. Varma, D. Stiliadis: Hardware Implementation of Fair Queueing Algorithms for Asynchronous Transfer Mode Networks. IEEE Communication Magazine (1997)

6. D. Stiliadis, A. Varma: A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms. IEEE Infocom '97 proceedings (1997)
7. J.C.R. Bennet, D. C. Stephens, H. Zhang: High Speed, Scalable and Accurate Implementation of Packet Fair Queueing Algorithms in ATM Networks. IEEE Communication Magazine (1997)
8. http://www.cs.cmu.edu/~cheeko/wf2q+/