Extending WF^{2}Q+ to Support a Dynamic Traffic Mix

Paolo Valente
Scuola Superiore S. Anna, Pisa, Italy
pv@gandalf.sssup.it

Abstract

WF^{2}Q+ is a packet scheduler providing optimal QoS guarantees at a low computational complexity. It allows a fraction of the total link capacity to be reserved to each packet flow to transmit, and it guarantees to each flow the minimum possible deviation with respect to its reserved service over any time interval.

WF^{2}Q+ has been defined assuming that the set of the packet flows to transmit is fixed and known at system design time. Unfortunately, such assumption is widely violated in many systems, such as Web servers or Internet routers. In this paper we propose a general scheme for extending WF^{2}Q+ to support also the case where the set of packet flows to transmit is unknown beforehand and varies over time. The scheme preserves the service guarantees provided by WF^{2}Q+, and allows different tradeoffs to be realized between computational complexity and system responsiveness to changing traffic mixes.

After investigating the pros and cons *of the possible solutions based on such general scheme, we present a simple and efficient algorithm for enabling WF^{2}Q+ to support a dynamic traffic mix.*

Keywords: Quality of Service, Resource Reservation, Computational Complexity.

1. Introduction

Packet scheduling algorithms at network nodes play a critical role in providing Quality of Service (QoS) guarantees in time-sensitive network applications.

A widely deployed model for providing QoS guarantees at a network node is the Resource Reservation one: for each outgoing link, a fraction (*share*) of the link capacity is reserved to each of the packet flows[1] served by the link. Possibly each flow must pass an Admission Control test before being accepted and transmitted through the link. Given any time interval during which a flow is continuously backlogged, we define as *reserved service* for the flow over such time interval, the minimum number of bits of the flow that *should* be transmitted according to its reserved share of the link capacity.

We said that a packet flow *should* receive its reserved service for the following reason. In a real system, a transmission link can transmit only one packet at a time, and the transmission of each packet can not be interrupted. It is then easy to prove that the minimum deviation from the reserved service achievable by any packet scheduler in a real system is roughly equal to twice the maximum packet size [4].

To the literature, the only two existing work-conserving[2] schedulers guaranteeing to each flow the minimum possible deviation from its reserved service over any time interval are Worst-case Fair Weighted Fair Queueing (WF^{2}Q) [3], and Worst-case Fair Weighted Fair Queueing Plus (WF^{2}Q+) [4]. Defined N as the number of packet flows sharing a common outgoing link at a network node, both WF^{2}Q and WF^{2}Q+ have $O(\log N)$ worst-case computational complexity per packet transmission [13].

Furthermore, both are based on the Proportional Share (PS) model: each flow is assigned a weight and, at any time instant t in which it is backlogged, it should *ideally* receive a share of the link capacity equal to the ratio between its weight and the sum of the weights of the flows backlogged at time t. This is exactly the service provided by a perfectly fair ideal system, called Generalized Processor Sharing (GPS) server [1]. The ideal PS service distribution provided by the GPS server is obviously unfeasible in a real system, but it is commonly used as a reference service distribution for practical packet schedulers.

A backlogged flow receives the minimum possible share of the link capacity if all the other flows are backlogged too. As a consequence, the Proportional Share model provides a natural implementation of the Resource Reservation one, where the reserved share of a flow coincides with the ratio between its weight and the sum of the weights of all the flows served by the link.

WF^{2}Q is based on the internal on-line simulation of the

[1] Classification issues are out of the scope of this paper, we just assume that packet flows be defined is some meaningful way.

[2] I.e. which never leave the link idle if there are pending packets.

service provided by the GPS server, and it guarantees the minimum possible per-flow deviation with respect to the simulated GPS server. As such, in addition to the service guarantees also provided by WF^{2}Q+, WF^{2}Q guarantees the minimum possible deviation from a perfect PS service. Said in other words, besides guaranteeing the minimum possible deviation from the reserved service as WF^{2}Q+, WF^{2}Q also guarantees the minimum possible deviation from a perfectly fair re-distribution of the excess bandwidth when not all the flows are backlogged.

Such higher service accuracy is paid with the computational cost of internally simulating the GPS server. In contrast, WF^{2}Q+ is based on the internal simulation of a more approximated, but cheaper to simulate system. For this reason, WF^{2}Q+ has the same asymptotic $O(\log N)$ computational complexity of WF^{2}Q, but with smaller constants.

Furthermore, an approximated implementation [12] of WF^{2}Q+ with $O(1)$ complexity (in the number of flows) has been devised to achieve high efficiency in high speed applications. Although not achieving the minimum possible deviation, such approximated implementation still guarantees a tight $O(1)$ per-flow deviation with respect to the reserved service.

As a conclusion, in all the applications where perfect fairness is not a major issue, whereas computational complexity is, WF^{2}Q+ constitutes the best option to achieve deterministic QoS guarantees. We will provide a wider comparison between WF^{2}Q+ and other existing schedulers in the section on related work.

1.1. The problem of the dynamic traffic mix

Both WF^{2}Q and WF^{2}Q+ have been defined assuming that the whole set REF of the N flows served by a link is fixed and known at system design time. But consider a popular Web-site or a core Internet router. In both systems, the set of competing flows can vary dramatically over a week, or even during each day.

Hence, which is exactly the static set REF of competing flows to consider? How can we assess its composition at system design time? Finally, is the same set compliant with the actual traffic mix at any time instant of the system lifetime?

The problem is that a static set of packet flows is not well suited for describing the dynamic nature of the packet traffic in a Web-site or in a network router.

This is not an issue in case of WF^{2}Q. According to what previously said, all that matters for simulating the service provided by a GPS server is knowing the set of the flows backlogged in the server at any time instant. For this reason, the set REF constitutes only a descriptive artifact in WF^{2}Q, and it can be trivially assumed to include all the possible flows that will be served during the system lifetime.

On the contrary, WF^{2}Q+ does need to know REF to compute its schedule. More precisely, as we will show, it needs to know the sum of the weights of all the flows in REF. In the end, differently from WF^{2}Q, WF^{2}Q+ must be properly extended to support also a dynamic traffic mix.

1.2. Contributions of this paper

In this paper we propose a *general scheme* for extending WF^{2}Q+ to support a dynamically changing traffic mix, still preserving all the service guarantees provided by the original scheduling algorithm. The main idea behind such scheme is: 1) defining, by mimicing the tracking of the set of the backlogged flows performed by WF^{2}Q, a dynamic set $REF(t)$, which provides at any time instant an approximation *by excess* of the set of the backlogged flows, and 2) substituting REF with $REF(t)$ in all the formulas used by WF^{2}Q+ to compute its schedule.

The scheme allows different tradeoffs to be realized between computational complexity and responsiveness to changing traffic mixes. We investigate the possible solutions by comparing their performance against the one of WF^{2}Q (the analysis is only qualitative in nature; a simulation testbed for carrying out a quantitative analysis is under preparation, as explained in the last section) . Finally, basing upon the outcomes of such analysis, we define Tick-driven Removal (TdR), a simple and efficient algorithm for enabling WF^{2}Q+ to support a dynamic traffic mix.

As we will show, both TdR and all the other possible solutions analyzed in this paper are compliant with *any* higher level resource reservation or admission control policy.

1.3. Organization of the paper

This paper is organized as follows. In the next section we make a brief survey of related work. In Section 3, we provide an overview of WF^{2}Q+. Then in Section 4 we present our general scheme for extending WF^{2}Q+, plus an analysis of the possible solutions based on such scheme. In the same section we present the algorithm TdR and its properties.

2. Related work

To the best of our knowledge, the issues related to supporting a dynamic traffic mix with WF^{2}Q+ has never been targeted.

In contrast, apart from WF^{2}Q, other scheduling algorithms with $O(\log N)$ complexity and able to support a dynamic traffic mix have been devised, such as Self Clocked Fair Queueing [6], and Start Time Fair Queueing [7]. Unfortunately, all them exhibit $O(N)$ per-flow deviation with respect to the reserved service, where N can be in the order

of the thousands in many applications, such as popular Web sites.

Finally, several schedulers with $O(1)$ complexity have been proposed to achieve high efficiency in high speed applications [8, 10], but all them exhibit $O(N)$ or, worse yet, unbounded per-flow deviation with respect to the reserved service.

3. WF^{2}Q+

Consider a system consisting of a network node in which N flows share a common outgoing link with time varying capacity $R(t)$. We will call such system the *real system* when we will need to distinguish it from the ideal system defined in Subsection 3.2. We define as REF the set of the N flows. We say that a packet *has arrived* in the system when its last bit has arrived in the system, we call packet *arrival time* the time at which this happens. Similarly, we say that a packet *departs* from the system when its last bit is transmitted by the system, and we call packet *finish time* the time at which it happens.

Each flow has a packet FIFO queue associated with it, holding the flow own backlog. We say that a flow is *backlogged* if it owns packets not yet (completely) transmitted, otherwise we say that the flow is *idle*. We define $B(t)$ as the set of the flows backlogged at time t. Finally, we define backlogged/idle period for the $i-th$ flow a time interval during which the flow is continuously backlogged/idle.

We define $W(t) \equiv \int_o^t R(\tau) \cdot d\tau$ as the *total amount of service* – bits transmitted – provided by the system during $[0,\, t]$. Given a generic function $f(t)$ of the time, we will use the notations $f(t^-)$ and $f(t^+)$ to refer to the value assumed by the function immediately before time t, and immediately after time t. Furthermore, given two time instants t_1 and t_2 such that $t_1 \leq t_2$, we define $f(t_1,\, t_2) \equiv f(t_2) - f(t_1)$. Finally, we define $df(t) \equiv f(t,\, t+dt)$, where dt is the length of an arbitrarily short time interval.

Before introducing WF^{2}Q+, we give in the next subsection a formal definition both of the PS paradigm, which WF^{2}Q+ adheres to, and of an index that measures the per-flow deviation of a scheduler with respect to the reserved service. In the successive subsection we will describe the ideal system internally simulated by WF^{2}Q+. We will finally show WF^{2}Q+ and its service properties in the last subsection.

3.1. PS paradigm and Worst-case Fair Index

Each flow i has a positive number ϕ_i assigned to it, namely its *weight*. Defined

$$\Phi_{TOT} \equiv \sum_{j \in REF} \phi_j \tag{1}$$

as the sum of the weights of all the N flows in REF, we define $\frac{\phi_i}{\Phi_{TOT}}$ as the *reserved share* of the $i-th$ flow.

In some papers, as e.g. [3, 4], Φ_{TOT} is assumed to be no higher than 1. Without entering into details, this assumption would simplify the formulas used in the definition of the time guarantees as well as of the algorithm itself. In all these formulas, Φ_{TOT} could be replaced by 1. Unfortunately, assuming $\Phi_{TOT} \leq 1$ has two negative consequences.

First, the weight of a flow happens to coincide with the reserved share of the flow if $\Phi_{TOT} = 1$, or is a lower bound of the reserved share if $\Phi_{TOT} < 1$. Hence, the weights assume an *absolute* meaning. On the contrary, we say that the weights have a *relative* meaning if the the *only* implication of assigning weights ϕ_i and ϕ_j to flows i and j is that the *ratio* between the reserved shares of the two flows is $\frac{\phi_i}{\phi_j}$. Assuming a relative meaning for the weights is very practical in many applications where achieving fairness is the only goal. Usually in these applications one does not want to care about absolute values, or about numerical problems related to meeting the $\Phi_{TOT} \leq 1$ constraint.

Second, systems with a high variability of the number of backlogged flows, as e.g. Web or Video-on-Demand servers may undergo a coarse bandwidth distribution. In these systems, the sum of the weights of the backlogged flows may be much lower than 1 during some time periods. During these periods each backlogged flow would be guaranteed a reserved share much lower than the share that it should enjoy according to the PS paradigm. Said in other words, the excess bandwidth would not be guaranteed to be redistributed in proportion to flow weights, as one would expect in an accurate PS system.

Both problems can be solved by removing the constraint $\Phi_{TOT} \leq 1$ and adopting the scheme proposed in this paper. As shown in Section 4, weights can be used with a relative meaning, and a fine PS bandwidth distribution can enforced against any traffic mix. Hence, in the rest of this paper, we assume no constraint on Φ_{TOT}.

According to what is said in the introduction, over any time interval $[t_1,\, t_2]$ during which it is continuously backlogged, the $i-th$ flow is expected to receive the following *reserved service* $W_{i,\,res}(t_1,\, t_2)$ (measured in number of bits transmitted)

$$W_{i,\,res}(t_1,\, t_2) \equiv \frac{\phi_i}{\Phi_{TOT}} \cdot W(t_1,\, t_2) \tag{2}$$

where $W(t_1,\, t_2)$ is the total amount of service provided by the system during $[t_1,\, t_2]$.

The following index, called Worst-case Fair Index (WFI) [3], can be used to measure the per-flow discrepancy between the service provided by a scheduling algorithm and the reserved service:

Worst-case fair Index (WFI). A scheduling algorithm

A is characterized by a Worst-case Fair Index WFI_i^A for the $i-th$ flow, if, given any time interval $[t_1,\, t_2]$ during which the $i-th$ flow is continuously backlogged,

$$W_i^A(t_1,\, t_2) \geq \frac{\phi_i}{\Phi_{TOT}} \cdot W(t_1,\, t_2) - WFI_i^A$$

where $W_i^A(t_1,\, t_2)$ is the amount of service provided by the scheduling algorithm A to the $i-th$ flow during $[t_1,\, t_2]$.

As we will show, WF^{2}Q+ has the smallest WFI among all packet schedulers. Such result is achieved by internally simulating the service provided by a special ideal system, and continuously serving packets so as to stay as close as possible to the service provided by the ideal system. We provide an overview of such ideal system in the next subsection.

3.2. Shaped-Rate Proportional Server (S-RPS)

The main reason why WF^{2}Q+ has the smallest WFI among all packet schedulers is because it closely approximates the service provided by an ideal system, that we will call Shaped-Rate Proportional Server (S-RPS), and that has the following WFI

$$WFI_i^{S-RPS} = (1 - \frac{\phi_i}{\Phi_{TOT}}) \cdot L_{i,\,max} \;\; \forall i \qquad (3)$$

where $L_{i,\,max}$ is the maximum length of the packets of the $i-th$ flow. We will progressively introduce all the elements needed to show how a S-RPS works, and, as soon as possible, we will show all the concepts through a graphical example. We will provide neither all the details nor any proof. The interested reader is referred to [5].

The S-RPS is a work-conserving ideal fluid system, i.e. a system that: 1) is never idle if there are backlogged flows, 2) can serve more than one flow at the same time, and can provide each flow with a different *service rate*. We define as capacity $R(t)$ of a S-RPS, the total amount of service provided by the system at time t.

In a S-RPS each flow i is associated with a *flow potential* $P_i(t)$, which measures – in the *special* way shown below – the *normalized* (i.e. divided by its weight) amount of service received by the flow ($P_i(0) = 0\,\forall i$):

$$P_i(t+dt) \equiv \begin{cases} P_i(t) + \frac{dW_i^{S-RPS}(t)}{\phi_i} & \text{if } i \in B(t) \\ \max\left[P_i(t),\, P(t)\right] & \text{otherwise} \end{cases} \qquad (4)$$

where $P(t)$ is a special function called *(S-RPS) system potential*. We will soon show the form of such function and the reason for the artificial *pushing up* of the idle flow potentials.

The S-RPS serves the backlogged flows so as to tend to *equalize* their potentials. In formulas, defined $C(t) \subseteq B(t)$ as the subset of the backlogged flows whose potential is minimum among all the backlogged flows, i.e.

$$C(t) \equiv \{i \in B(t) \,|\, P_i(t) = \min_{j \in B(t)} P_j(t)\} \qquad (5)$$

a S-RPS provides the following service to each flow:

$$dW_i^{S-RPS}(t) = \begin{cases} \frac{\phi_i}{\sum_{j \in C(t)} \phi_j} \cdot dW(t) & \text{if } i \in C(t) \\ 0 & \text{otherwise} \end{cases} \qquad (6)$$

where $dW(t) = R(t) \cdot dt$ is the total amount of service provided by the system during $[t,\, t+dt]$.

To show a graphical example, suppose for the moment that the system potential is just a linear function of $W(t)$, growing at the minimum possible slope with which flow potentials can grow, i.e.

$$P(t) = \frac{1}{\Phi_{TOT}} \cdot W(t) \qquad (7)$$

(we will see the complete function after the example). Figure 1 shows a possible evolution of the flow potentials and of the system potential in a S-RPS with constant capacity $R(t) = R$. The S-RPS serves two flows, both with weight 1. Each arriving packet is depicted in Figure 1.A as a rectangle: the projection on the x axis of the left corner of the rectangle represents the packet arrival time, while the length of the base represents the time needed to transmit the packet at full system speed.

According to (6), the first flow is served at maximum speed during $[a_1,\, a_2)$, as shown in Figure 1.C (the areas represent the amount of service received by the flows). Hence, according to (4) and (7), $P_1(t)$ grows faster than the system potential. The potential of the second flow is pushed up through the system potential during its idle period $[a_1,\, a_2]$, i.e. $P_2(t) = P(t)\;\forall t \in [a_1,\, a_2]$. Hence $P_2(a_2) = P(a_2)$ when the first packet of the second flow arrives. Since $P_2(a_2) < P_1(a_2)$, then, according to (6), the service of the first flow is suspended until both potentials become equal at time b. After time b, both flows receive the same service rate, until the packet of the first flow is completely transmitted at time F_1. The first packet of the second flow is instead completed at time F_2.

The figure also shows the consequences of the artificial pushing up of the idle flow potentials. During $[a_1,\, a_2)$, $P_2(t)$ grows as if the second flow was backlogged and was receiving the minimum service guaranteed by (6). If $P_2(t)$ was not pushed up during $[a_1,\, a_2)$, then, during the successive backlogged period and according to the equalizing policy (6), the second flow would have recovered the service lost while it was idle, which is not provided by the PS guarantee (2). Finally, it is easy to show that such service recovery may cause the PS guarantee (2) to be violated for some flows. On the contrary, pushing up the potential of each idle

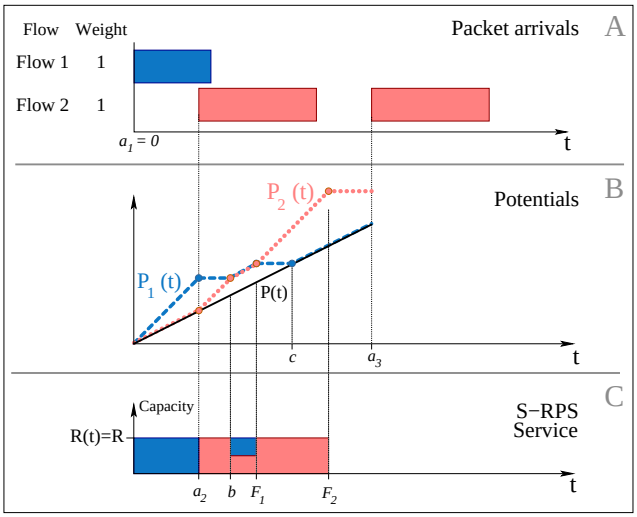

Figure 1. Evolution of the potentials in a S-RPS.

flow (if it becomes lower to the system potential) guarantees that the flow be treated as if it had received a *fictitious* extra amount of service, such that the flow never happens to have received less service than the minimum guaranteed in case it was continuously backlogged.

As a conclusion, if the flow potentials are guaranteed to be *never* lower than the system potential also while they are backlogged, and if the system potential never grows slower than any backlogged flow potential, we can give the following physical meaning to the system potential: $P(t)$ represents the minimum normalized amount of service guaranteed to each flow during $[0,\ t]$, given by the sum of two components: the sum of the minimum amount of normalized service *actually* guaranteed to the flow during each of its backlogged periods, plus the sum of the minimum amount of normalized service that would have been guaranteed to the flow if it *had been* backlogged during each of its idle periods.

The properties of the S-RPS have been proven assuming that the above physical meaning of the system potential holds [5]. Hence, it is crucial to note that, if such physical meaning is violated, the proofs of the service guarantees provided by the S-RPS, and hence, by WF^{2}Q+, are broken too.

There is a complementary problem. Consider again Figure 1. Since the first flow accumulated more than the guaranteed service during $[a_1,\ a_2)$, then, during $[a_2,\ b)$, it pays back for the *extra service* previously received. Basing upon this observation, it is easy to prove that, if a flow could receive an unbounded amount of extra service during some time interval, then it might suffer for a complete service denial proportional to the amount of extra service previously received.

For this reason, the service provided by the S-RPS is *shaped* through the following rule: the S-RPS is allowed to start serving a packet only if such packet is *eligible*, i.e. if the potential of the corresponding flow is no higher than the system potential.

For example, in Figure 1 the first packet of each of the two flows is eligible as it arrives, whereas the second packet of the second flow is not eligible when it arrives at time a_3. This fact causes a problem: since *no* packet is eligible at time a_3, the system risks to be non work-conserving.

The problem is solved by using a slightly more complex system potential function than (7). To define the complete function, we need to introduce the following quantities. Given the $k-th$ packet p_i^k of the $i-th$ flow, we define *start potential* S_i^k and *finish potential* F_i^k of p_i^k, as the value assumed by the S-RPS system potential when p_i^k starts to be served and when p_i^k is completed, respectively. It has been proven in [4] that

$$\begin{aligned} S_i^k &= \max(P(a_i^k),\ F_i^{k-1}) \\ F_i^k &= S_i^k + \frac{L_i^k}{\phi_i} \end{aligned} \tag{8}$$

where a_i^k and L_i^k are, respectively, the arrival time and the length of the packet p_i^k. The complete formula for the system potential is

$$P(t+\tau) = \max\{P(t) + \frac{W(t,t+\tau)}{\Phi_{TOT}},\ \min_{i \in B(t+\tau)}\{S_i^{h(t)}\}\} \tag{9}$$

where $S_i^{h(t)}$ is the start potential of the packet $p_i^{h(t)}$ at the head of the queue of the $i-th$ flow at time t. Consider again Figure 1. According to (9), at time a_3^+ the system potential *jumps* to $P_2(a_3)$, thus guaranteeing the system to remain work-conserving.

We can finally enunciate the fundamental property that we will preserve when extending WF^{2}Q+ in the next section:

Property 1 *Suppose that, whenever the value of the system potential (9) is equal to the lowest flow potential in the system, the system potential grows no faster than how the potentials of the flows currently under service grow. Then the physical meaning of the system potential (9) is preserved and (3) holds.*

3.3. Definition and properties of WF^{2}Q+

WF^{2}Q+ is based on the internal simulation of a *corresponding* S-RPS, i.e. of a S-RPS with the same capacity of the real system at any time instant, and receiving the same packet arrival pattern as the real system.

The following three properties are exploited in the definition of WF^{2}Q+: 1) given any packet p_i^k, its start and finish potentials S_i^k and F_i^k in the corresponding S-RPS can

be efficiently computed using (8), 2) a packet p_i^k is eligible at time t if and only if $S_i^k \leq P(t)$, and 3) it is easy to prove that, at any time instant, the eligible packet with the smallest finish potential is the next packet to finish in the corresponding S-RPS if no other packet is to arrive [4]. The definition of WF^{2}Q+ follows.

Worst-case Fair Weighted Fair Queueing Plus (WF^{2}Q+). Whenever the link is available to transmit a new packet, transmit the eligible packet with the minimum finish potential. Ties are broken arbitrarily.

As a consequence, simulating the corresponding S-RPS means just computing its system potential according to (9) and using it to:1) compute packet start and finish potentials through (8), and 2) filter eligible packets[3].

It has been proven in [5] that approximating as closely as possible the packet finish order of the corresponding S-RPS preserves a bounded per-flow lag, equal to L_{max}, with respect to the service provided by the corresponding S-RPS. Besides, for the same reasons shown for the S-RPS itself, the eligibility constraint bounds to $(1 - \frac{\phi_i}{\Phi_{TOT}}) \cdot L_{i,\,max}$ the maximum amount of extra service provided to each flow by WF^{2}Q+ with respect to the minimum guaranteed by the corresponding S-RPS according to the system potential (9). It has been proven in [3] that, thanks to such bounds, and if (3) holds, then

$$WFI_i^{WF^2Q+} = (1 - \frac{\phi_i}{\Phi_{TOT}}) \cdot L_{i,\,max} + L_{max} \qquad (10)$$

which is the minimum possible WFI achievable in a real system [4].

4. Supporting a dynamic traffic mix

In this section we will always refer to a pair of system made of a real system in which WF^{2}Q+ is used to schedule packets, and the corresponding S-RPS.

As shown in the introduction, the static set REF is not well suited for describing the dynamic nature of the traffic mix in a network router or in a Web server. We also showed that this fact is not a problem in case of WF^{2}Q, where REF constitutes only a descriptive artifact.

On the contrary, REF has an operative meaning in WF^{2}Q+, because it determines the value of Φ_{TOT} as computed in (1), and, according to (9), Φ_{TOT} must be known to track the S-RPS system potential.

To solve the problem, we propose to *mimic* the backlogged flows tracking performed by WF^{2}Q. First, assume, as done for WF^{2}Q, that REF is the set of all the possible flows that will be served during the system lifetime. Then use a dynamic set $REF(t) \subseteq REF$ instead of REF when computing the sum of the weights of the flows served by the system. I.e. use the dynamic quantity

$$\Phi_{TOT}(t) \equiv \sum_{j \in REF(t)} \phi_j$$

instead of $\Phi_{TOT} \equiv \sum_{j \in REF} \phi_j$ in (9). Finally, update $REF(t)$ according to the following general scheme:

General scheme for updating REF(t).

1. At system start up $REF(t) = \emptyset$.
   (a) If a flow not belonging to $REF(t)$ becomes backlogged at time t, it is inserted in $REF(t)$.

   (a) A flow can be removed from $REF(t)$ only if it is idle in the corresponding S-RPS. The removal of a flow from $REF(t)$ may be delayed for a maximum time interval ΔT from the time instant in which the flow becomes idle in the corresponding S-RPS. If a flow idle in the corresponding S-RPS becomes backlogged in the corresponding S-RPS before being removed from $REF(t)$, its removal is canceled.

According to the above scheme, the set $REF(t)$ is a sort of approximation *by excess* of the set of the flows backlogged in the corresponding S-RPS: flows are allowed to be removed *lazily* from $REF(t)$.

Before showing possible algorithms compliant with such a scheme, we want to prove that the scheme is well suited for preserving the service guarantees provided by WF^{2}Q+. According to Definition 4, a new flow can be added to $REF(t)$ as soon as it becomes backlogged. Hence $\Phi_{TOT}(t)$ can be immediately increased, and ultimately the minimum slope of $P(t)$ can immediately decrease in (9). This does not cause any problem to the simulated S-RPS in meeting Property 1. On the contrary, according to (6), a lower system potential slope makes the hypothesis of Property 1 easier to be guaranteed.

With regard to removals, it is dangerous to remove a flow from $REF(t)$ if the flow is still backlogged in the corresponding S-RPS. Suppose to remove a flow from $REF(t)$, and hence to decrease $\Phi(t)$ as soon as the flow becomes idle in the real system. Consider then Figure 2, where two flows with unit weight present one packet each at time 0. Figures 2.B and 2.C show, respectively, the service provided by WF^{2}Q+ and the evolution of the potentials.

Both flows are inserted in $REF(t)$ at time 0, hence $\Phi_{TOT}(0^+) = 2$. According to (6), (4), and (9), both the system potential and the potentials of the two flows grow at the same rate.

At time f_1 the first flow becomes idle in the real system. It is then immediately removed from $REF(t)$, hence

[3]For the sake of precision, the set $B(t)$ actually used in (9) is not the set of the flows backlogged in the corresponding S-RPS, but the set of the flows backlogged in the real system. However, it has been shown in [5] that this fact does not affect the hypothesis in Property 1, and that both systems are still guaranteed to be work-conserving.

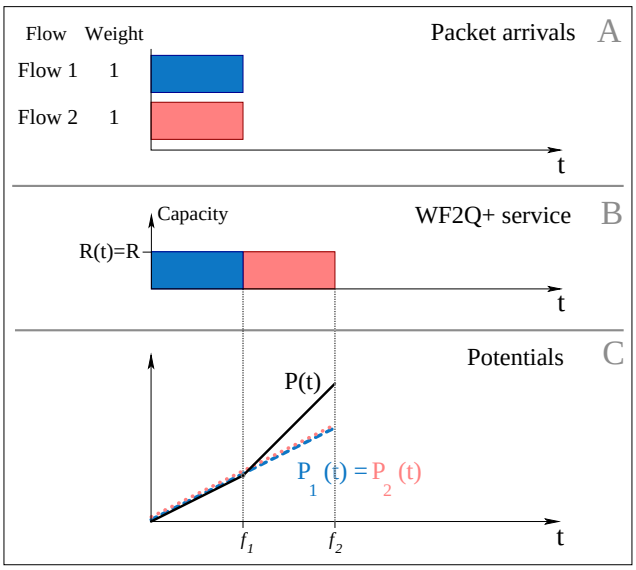

Figure 2. Consequences of the early removal of a flow from $REF(t)$.

$\Phi_{TOT}(f_1^+) = 1$, and the system potential starts growing at twice its previous slope. But both flows are still receiving service in the corresponding S-RPS, hence the slope of their potentials remains unchanged at time f_1^+. As a conclusion, during $[f_1,\ f_2]$ the system potential grows at twice the slope of the flow potentials. Hence it *violates* the hypothesis of Property 1, through which the guarantee (3) has been proven.

On the contrary, if a flow is removed from $REF(t)$ only after becoming idle in the corresponding system, then it can never happen that $\Phi_{TOT}(t) \leq \sum_{j \in C(t)} \phi_j$. Hence, according to (9) and (6), $P^{S-RPS}(t)$ will never violate the hypothesis of Property 1, hence the service guarantees (3) and (10) continue to hold.

As a conclusion, the scheme 4 allows no service guarantees to be broken. But, how can a flow be asserted to be idle? As previously said, WF^{2}Q+ carries out the simulation of the corresponding S-RPS by just tracking the evolution of its system potential. Hence, the system potential constitutes the only available source of information on the service provided by the corresponding S-RPS.

Unfortunately, the system potential does not contain information on the actual service provided to the flows, but only on the minimum service guaranteed. As a consequence, defined as *unbacking potential* of the $i-th$ flow the finish potential of the last packet of the $i-th$ flow arrived up to time t, the system potential provides only the following sufficient, but not necessary, condition for a flow to be idle in the corresponding S-RPS: if the system potential is no lower than the unbacking potential of the $i-th$ flow at time t, then, at time t, the $i-th$ flow is certainly idle in the corresponding S-RPS. We say that the $i-th$ flow is *disposable* at time t if its unbacking potential is no higher than the system potential at time t. For example, the first flow becomes disposable at time c in Figure 1, although it was already idle at time F_1.

The minimum worst-case delay ΔT in updating $REF(t)$ – as defined in Definition 4 – achievable by any algorithm that removes only disposable flows from $REF(t)$, is equal to the maximum time interval ΔT_{disp} ranging from the time instant in which a flow becomes idle in the corresponding S-RPS to the time instant in which it becomes disposable. More generally, basing upon the disposable attribute and using a more or less *lazy* removal strategy, the desired tradeoff between computational efficiency and system *responsiveness* to changing traffic mixes can be realized.

Several tradeoffs will be discussed the next subsection. Finally, exploiting the considerations made in such subsection, a simple and efficient algorithm for updating $REF(t)$ will be defined in the last subsection.

4.1. Possible solutions

As shown in [13], in a time interval during which $\Phi_{TOT}(t)$ is much larger than the sum $\Phi_{back}(t)$ of the weights of the backlogged flows, WF^{2}Q+ may provide an unfair/bursty service. Hence, achieving a low value for ΔT in Definition 4 can greatly improve the short term fairness and the service smoothness of WF^{2}Q+.

The quickest strategy for updating $REF(t)$ by the means of the disposable attribute is removing all the disposable flows from $REF(t)$ upon each system potential update. Hereafter we will shortly refer to such updating strategy as the quickest strategy.

To evaluate the effectiveness of the quickest strategy and of other less aggressive strategies for updating $REF(t)$, we must consider that WF^{2}Q can be used to provide a perfectly fair/smooth service over any time interval [13]. On the contrary, as previously shown, even the quickest strategy can not guarantee a worst-case delay lower than ΔT_{disp} in updating $REF(t)$. As a consequence it can not prevent $\Phi_{TOT}(t)$ from being much larger than $\Phi_{back}(t)$ – and hence WF^{2}Q+ from providing an unfair/bursty service – during some time intervals with length ΔT_{disp}.

Hence, using WF^{2}Q+ instead of WF^{2}Q may still be advantageous if the former can be implemented at a lower complexity than WF^{2}Q. In contrast, aggressive strategies for updating $REF(t)$ can lead to high computational complexity. Especially, the quickest strategy involves $O(N)$ worst-case flow removals per system potential update, because, as shown for the GPS server in [13], packet finish times can be arbitrarily slightly skewed in a fluid system.

$\Phi_{TOT}(t)$ could be however updated at $O(\log N)$ worst-case cost per packet service. Consider Logarithmic-WF^{2}Q (L-WF^{2}Q), the $O(\log N)$ implementation of WF^{2}Q shown in [13]. To simulate the service provided by the GPS server, L-WF^{2}Q maintains a special tree with $O(\log N)$ depth. It is easy to show that, using a similar data structure, it is possible to compute at $O(\log N)$ cost the new value of $\Phi_{TOT}(t)$ after removing $O(N)$ disposable flows from $REF(t)$.

But, even exploiting such data structure, the quickest strategy would however cause WF^{2}Q+ to have an additional running time per packet service no lower than the one incurred by L-WF^{2}Q to simulate the GPS server. Then we must consider strategies more tilted toward a lower computational complexity.

The additional computational complexity of L-WF^{2}Q with respect to WF^{2}Q+ is due to the fact that, each time a new packet arrives and each time the next packet to transmit must be chosen, L-WF^{2}Q must read/update at most three fields for each node along a path from the root to a leaf of the above mentioned $O(\log N)$ depth tree. Then we can deduce that an extended version of WF^{2}Q+ supporting a dynamic traffic mix can be implemented at a significantly lower cost than WF^{2}Q only if the algorithm for updating $REF(t)$ removes from $REF(t)$ no more than 2-3 disposable flows per packet service/arrival. A low complexity algorithm that enjoys such property is presented in the next subsection.

Before leaving this subsection, it is worth noting that all the above outlined solutions – as well as the algorithm presented in the next subsection – allow $REF(t)$ to be updated automatically and independently of the possible service policies implemented on top of the scheduler. As such, they are compliant with *any* higher level resource reservation or admission control policy.

4.2. An efficient solution

Basing upon the considerations made in the previous subsection, and assuming that the system is equipped with a system clock able to raise a periodic *tick* interrupt, we can define the following simple and efficient algorithm for updating $REF(t)$:

Algorithm Tick-driven Removal (TdR). As soon as a flow becomes idle in the real system, insert it in a special *idle queue* – ordered by flow unbacking potentials. At each *tick* of the system clock, check if the flow at the head of the idle queue is disposable and, if so, remove it both from $REF(t)$ and from the idle queue. If a flow in the idle queue becomes backlogged in the real system again, remove it from the idle queue.

At each clock tick it is possible to check at $O(1)$ cost whether or not the flow at the head of the idle queue is disposable, by just comparing its unbacking potential with the current value of the system potential. Moreover, insertions and extractions from an ordered queue can be implemented at $O(\log N)$ cost with small constants [11]. Hence the TdR algorithm has $O(\log N)$ complexity per clock tick.

Furthermore, if T_{tick} is the the tick inter-arrival time, then, according to definition 4, it is easy to prove that a flow idle in the corresponding S-RPS is guaranteed to be removed from $REF(t)$ within a time interval

$$\Delta T = \max\left[\Delta T_{disp},\, \Delta T_{idle}\right] + N \cdot T_{tick}$$

where ΔT_{idle} is the maximum time interval from the time instant in which a flow becomes disposable to the time instant in which it becomes idle in the real system.

5. Conclusions and future work

We proposed a *general scheme* for extending WF^{2}Q+ to support also a dynamically changing traffic mix, still preserving all the service guarantees provided by the original scheduling algorithm. We investigated the different trade-offs allowed by the scheme between computational complexity and responsiveness to changing traffic mixes, and we compared the performance of the possible solutions against the one of WF^{2}Q. Finally, basing upon the outcomes of such analysis, we defined Tick-driven Removal (TdR), a simple and efficient algorithm for enabling WF^{2}Q+ to support a dynamic traffic mix.

We also showed that both TdR and all the other discussed solutions are compliant with *any* higher level resource reservation or admission control policy. To assess the actual effectiveness of our proposal, we still need to provide a quantitative comparison among the different algorithms – L-WF^{2}Q, WF^{2}Q+ with TdR and with the quickest strategy extension – in terms of computational overhead and fairness guarantees. To this aim, we are currently implementing them in a general purpose operating system (FreeBSD) as well as in the *ns-2* simulator.

References

[1] A. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control - the single node case", in *Proceedings of INFOCOM '92*, vol. 2, pp. 915-924, 1992.

[2] D. Stiliadis and A.Varma, "Rate-proportional flows: A general methodology for fair queueing algorithms", *IEEE/ACM Transactions on networking*, 1996.

[3] J. C. R. Bennett e H.Zhang, "WF^{2}Q: Worst-case fair weighted fair queueing", in *Proceedings of IEEE INFOCOM '96*, pp. 120-128, San Francisco, CA, 1996.

[4] J. C. R. Bennett e H.Zhang, "Hierarchical packet fair queueing algorithms", in *Proceedings of ACM SIGMETRICS '96*, pp. 143-156, 1996.

[5] D. Stiliadis and A. Varma, “A general methodology for designing efficient traffic scheduling and shaping algorithms", in *IEEE INFOCOM’97*, Japan, 1997.

[6] S. Golestani. “A self-clocked fair queueing scheme for broadband applications”, in *Proceedings of IEEE INFOCOM’94*, pages 636–646, Toronto, CA, 1994.

[7] P. Goyal, H.M. Vin, and H. Chen. "Start-time Fair Queueing: A scheduling algorithm for integrated services." In *Proceedings of the ACM-SIGCOMM 96*, pages 157-168, Palo Alto, CA, August 1996.

[8] M. Shreedhar and G. Varghese. “Efficient fair queueing using deficit round robin”, in *Proceedings of SIGCOMM’95*, pages 231–243, Boston, MA, 1995.

[9] S. Ramabhadran, J. Pasquale. “Stratified Round Robin: A Low Complexity packet Scheduler with Bandwidth Fairness and Bounded Delay”, in *Proceedings of ACM SIGCOMM’03*, 2003.

[10] C. Waldspurger. “Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management”, PhD thesis, MIT, 1995.

[11] T. H. Cormen, C. E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. The MIT Press, 1991.

[12] Stephens, D.C.; Bennett, J.C.; Zhang, H. “Implementing scheduling algorithms in high-speed networks” in *IEEE JSAC Special Issue on High Performance Switches/Routers*, 1999.

[13] P. Valente, “Exact GPS simulation with logarithmic complexity, and its application to an optimally fair scheduler”, *Proceedings of SIGCOMM’04*, 2004.