Design and analysis of a dynamic Weighted Fair Queuing (WFQ) scheduler

Gianmarco Panza, Matteo Grazioli, Filippo Sidoti

Abstract – **This work aims to analyse different design proposals for a dynamic Weighted Fair Queuing (WFQ) scheduler to provide Quality of Service (QoS) guarantees in general at IP level and specifically for the applications of the PHOENIX project.**
Several possible design options and configuration parameters have been investigated and studied for a dynamic scheduling discipline able to support QoS in a relative manner according to factors pre-assigned to each queue at a given interface.
Simulation results have demonstrated that this system can perform well in various network working conditions.
Finally, guidelines for a designer and a network operator have been carried out on the basis of the analysis of several collected simulation results.

I. INTRODUCTION

The PHOENIX project main goal is to effectively exploit the available bandwidth on wireless links (WLAN, UMTS, 4G…) that is dynamic by nature, providing optimised solutions for multimedia transmission over IP-based wireless networks. To reach this goal, it is proposed to develop a scheme offering the possibility to let the application world (source coding, ciphering) and the transmission world (channel coding, modulation) talk together over an IP protocol stack, so that they can jointly develop an end-to-end optimised wireless communication.

However, adequate Quality of Service (QoS) guarantees must be provided along the communication path throughout the network, both for the user data and the control/signalling information to be exchanged.

Advances in computing and network technologies have made it possible to support QoS guarantees in packet switching network. There are two approaches that were proposed in the literature, i.e. Integrated Service (IntServ) [1] and Differentiated Service (DiffServ) [2][3]. IntServ tried to ensure end-to-end and per flow QoS of data traffics. However, it is quite difficult to implement on a large scale, such as in the Internet. For this reason, DiffServ that only provides a limited number of service classes was then proposed. It aggregates the flows with the same QoS requirements, and serves the packets according to predefined traffic contracts. Instead of achieving per-flow QoS, DiffServ provide guarantees at an aggregate level.

Relative and absolute service differentiation are two categories of the DiffServ model. Relative DiffServ ensures the ratios of the quality level between classes, while absolute DiffServ absolute service guarantees. Obviously, the first is more straightforward to deploy as well as effective with proper resource provisioning in the concerned network.

There is a variant in this category, called proportional differentiation service [5].

Weighted Fair Queuing (WFQ) [4] is a scheduling discipline nowadays widely applied to QoS-enabled routers. In WFQ, single flows or traffic classes are served on the basis of the weight assigned to the related queue. The weight is determined according to the granted QoS parameters, such as service rate or delay. An absolute service rate can be easily achieved by assigning a fixed weight.

In this paper, we analyze the possible design options and configuration parameters of a Dynamic Weighted Fair Queuing scheduler (Dynamic WFQ), which is an extension of WFQ. The issued dynamic WFQ scheduler adjusts the weight of each class (queue) dynamically so that the delay differences between classes can be well controlled. Based on our model, the network operator can impose the ratio of the delays between the different classes, and maximize the network resources utilization. To simplify the description, in this paper we consider per-hop queuing delay only. Other QoS parameters and the characteristics of end-to-end delay are left for future work.

The remainder of this paper is organized as follows. Next section briefly explains the main achievements in the field, concerning both the static and the dynamic versions of WFQ. Then, a description of the work, together with the main simulation results and analysis are reported.

Finally, last section summarizes the main conclusions and paves the way for future developments.

II. WEIGHTED FAIR QUEUING DISCIPLINE

A. Static WFQ

Weighted Fair Queuing (WFQ) [4] offers fair queuing that divides the available bandwidth across queues of traffic based on weights. Each flow or aggregate thereof is associated with an independent queue assigned with a weight, so as to ensure that important traffic gets higher priority over less important traffic. In times of congestion the traffic in each queue (a single flow or an aggregate of them) is protected and treated fairly, according to its weight.

Arriving packets are classified into different queues by inspection of the packet header fields, including characteristics such as source and destination network or MAC address, protocol, source and destination port and socket numbers of the session or Diff-Serv-Code-Point

F. Sidoti is with WIND Telecomunicazioni S.p.A., Rome, Italy (e-mail: Filippo.Sidoti@mail.wind.it)
G. Panza is with CEFRIEL Network Systems Unit, Milan, Italy (e-mail: panza@cefriel.it)
M. Grazioli is with CEFRIEL Network Systems Unit, Milan, Italy (e-mail: grazioli@cefriel.it)

(DSCP) value. Each queue shares the transmission service proportionally to the associated weight. All traffic in the same class is treated indistinctly.

WFQ can certainly ensure satisfactory response time to critical applications, such as interactive and transaction-based ones, that are intolerant to performance degradation, in particular whether deployed in an Int-Serv architecture. In a Diff-Serv architecture, WFQ can be IP DSCP-aware. This means that it is able to detect higher priority packets marked with precedence and can schedule them faster, providing superior response time for these traffic aggregates.

In summary, from a technical point of view WFQ has three desirable properties. First, because it approximates GPS (General Processor Sharing) scheduler [4], it protects traffic of different queues from each other, which is fundamental in a service differentiation context. Second, traffic in a queue can obtain worst-case end-to-end queuing delay that is independent of the number of hops it traverses and of the behavior of traffic in the other queues. This allows networks of fair queuing schedulers to provide real-time performance guarantees. Third, it gives users an incentive to implement intelligent flow mechanisms at the end-system. A source is not required to send at a rate smaller than its currently allocated rate, however if it sends more than its fair share it can lose packets, so it has an incentive to match its flow to the currently available service rate.

B. *Dynamic WFQ*

A Dynamic WFQ [5][6] is able to dynamically and consistently adapt the queue weights according to the time-variant amount of the incoming traffic and the pre-assigned target QoS.

The fundamental issue is to correlate the burstiness of the traffic with the weight value in order to achieve given delay and loss guarantees.

The measurement of the burstiness of the aggregate entering each queue could be realized by evaluating the resulting buffer dimension [5], which is somehow related to the worst-case delay experienced by packets in the queue.

For what concerns the QoS, we could apply a proportional relative model. Each queue has assigned a static parameter and the performance guarantees provided to the set of queues should be in line with the mutual ratio of the said parameters. For example, if the queue Qi and Qj have the parameters Pi and Pj respectively, with Pj=2*Pi, the QoS provided to Qj should be two times better then the one granted to Qi. Hence, no absolute QoS assurances are supported in this case.

As proposed in [5], where also preliminary results are reported, if in a given interval Tn, Bn represents the average buffer dimension of Qn, which is associated with the parameters Pn, the queue weights could be determined by the resolution of a linear system whose equations are of the form:

$$(Bi/Bj)*(Pi/Pj) = (Wi/Wj)$$

where, Wi and Wj are the weights to be assigned to the queues Qi and Qj, respectively.

III. WORK DESCRIPTION AND SIMULATION RESULTS

A *Traffic Description and Parameter Settings*

We considered H.263 video flows at different bit rates, ranging from 64 to 256 Kbit/s as mean value, generated by real traces [7] of video streaming and conferencing applications (see the table below for more details). By their nature of typical compressed video flows, the related bit rate is highly variable with a burstiness factor (peak to mean rate ratio) of even 10.

Encoder Input	176x144 pel (QCIF)
N° of pixels for Chrominance	88
Frame Rate	25 fps
Quantization Parameter	5
Pattern	IBPBPBPBPBP
Integer pel search window	15 pels

Table 1 - Characteristics of a considered H.263 compressed video flow

Specifically, the traffic aggregate is composed of 3 video streams at 64 Kbit/s and 12 video streams at 256 Kbit/s.

A FIFO scheduler fed with such an aggregate leads to a maximum delay of 10 ms for 99% of all packets over a 10 Mbit/s link.

This bound for the 99^{th} percentile of the delay at each single router interface is just an example, anyway it allows for an adequate QoS for the applications of the PHOENIX project (i.e. real-time multimedia applications).

Initially, we studied the performance granted to the service classes by a static WFQ scheduler with 4 queues, each one fed with the described traffic aggregate, and a 40 Mbit/s output link. With the goal to highlight the limitations of an actual static packet scheduler, which reveals lower performance in providing the desired QoS parameters than in an ideal case (such as in a GPS system). This means that it is necessary to allocate more resources, specifically more bandwidth, to a given queue in order to achieve the 10 ms delay bound for 99% of packets.

For example, we focused our attention on the third queue and varied the associated weight to achieve the target performance. We can think of 4 ordered weights, from the lowest (first queue) to the highest (fourth queue). The sum of the weights must be of course equal to one in all the conducted simulations.

We expect that the third queue requires a weight greater than 0.25, high enough to compensate for the approximations and limitations introduced by a real scheduler.

Subsequently, we investigated a dynamic version of WFQ and its performance.

In the next paragraph we report and analyze the simulation results, produced by means of OPNET Modeler tool by OPNET Technology Inc.. For the Dynamic WFQ we considered several mechanisms to measure the traffic and various configuration parameters even in different working conditions, in order to provide useful guidelines for the network designer and operator.

B. *Simulation Results*

In the first simulation scenario the weights of the 4 queues were all set to 0.25. The graphs of figure 1 clearly have demonstrated the need to increase the third weight to

achieve the target performance in an actual static WFQ scheduler.

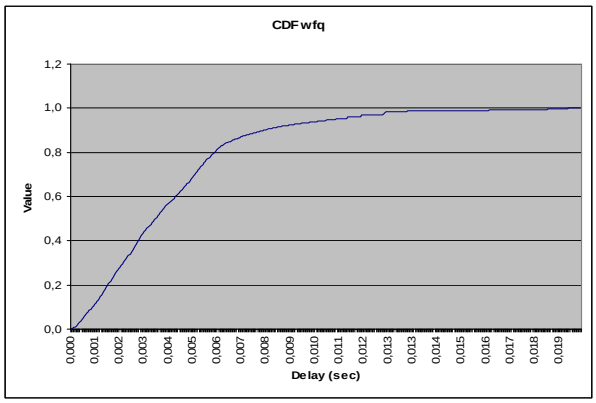

Figure 1 – CDF (Cumulative Distribution Function) of the 3rd queue with a 0.25 weight

The figure below depicts the CDF of the third queue delay with a bandwidth allocation dictated by weight values of 0.10, 0.18, 0.32 and 0.4 respectively, for the four queues. This means that in this scenario the additional bandwidth to be allocated was about 28% (the weight of the third queue was 0.32 instead of 0.25).

We realized that the mean delay of the analyzed queue was quite smaller than in the first case; nevertheless, the related weight had to be big enough to compensate for the impact of the worst-case behaviour of the static WFQ in particular when traffic bursts arrived. It can be noted that the CDF of the third queue, has higher value at lower delays with an area approximately equals to 0.99 around 10 ms as in the ideal case.

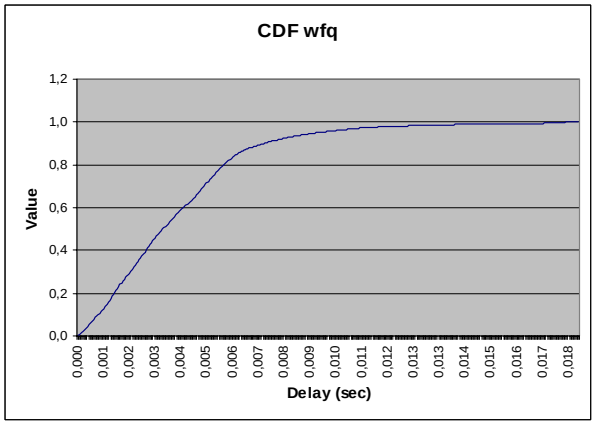

Figure 2 – CDF of the 3rd queue with a 0.32 weight

The following figure illustrates the relationship between the weight of the third queue and the incoming traffic aggregate rate, when the packet delay was approximately 10 ms. The graph was obtained considering the mean value of several measurements for the considered rate and varying the third queue weight, in order to obtain consistent data (each sample was gathered on a 40 ms interval).

By analysing such simulation results, it is clear how with a proper resource allocation, i.e. bandwidth, it is possible to assure stringent QoS guarantees, such as delay (and loss, with a proper buffer dimensioning).

However, figure 2 highlights a possibly considerable waste of resources to achieve the target service parameters when the traffic has a bursty nature.

For this reason, a dynamic WFQ was proposed [5] and widely investigated in our work.

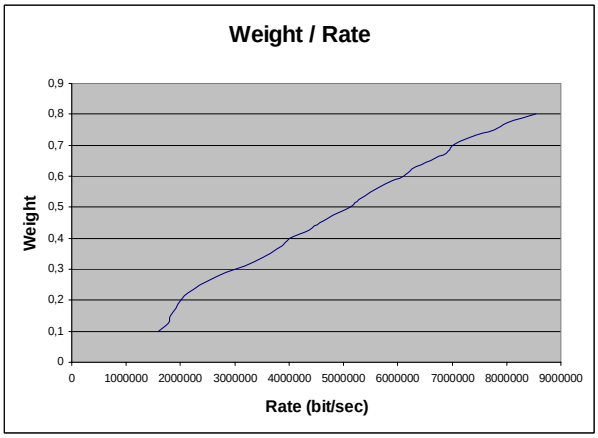

Figure 3 - Relationship between weight and aggregate rate to achieve the target QoS guarantees

As discussed in sect. II.B, the queue weights could be calculated according to such equations:

$(Bi/Bj)*(Pi/Pj) = (Wi/Wj)$

where, Wi and Wj are the weights to be assigned to the queues Qi and Qj. In our simulations, the parameters Pi were set to 1, 2, 3 and 4 respectively, for the four queues; hence, the second queue should have a delay guaranteed two times better than the first one and the fourth queue two times better than it.

The measurement interval, at the end of which the queue weights are updated, was initially fixed, in this first instance, to a quite small value, such as 40 ms, in order to well highlight the benefit of a dynamic scheduler.

The buffer dimension was determined according to a low-pass filter processing as follows:

$Bi(n)=k*Bi(n-i)+(1-k)*Bi_{ist}(n)$

Where Bi(n) represent the average buffer dimension at the n-th interval of Queue i and $Bi_{ist}(n)$ the instantaneous value for Queue I in the same interval. The factor k determines the width of the low-pass filter (the higher the value, the narrower the filter bandwidth).

Queue	Average Delay (ms)	99th percentile (ms)
1	4.8	22
2	4.2	17
3	3.5	11
4	3.2	7

Table 2 – Results with a 0.95 k filter

Queue	Average Delay (ms)	99th percentile (ms)
1	6.5	22.5
2	5.2	17.3
3	4.0	10.5
4	3.1	6.9

Table 3 – Results with an optimum filter (0.992 k)

We investigated the impact on the performance of a different low-pass filter. It is worthwhile to underline that the goal was to obtain relative QoS differentiation between the classes as much as possible accordingly to the associated quality parameters Pi. Absolute delay guarantees can be achieved with a proper resource provisioning and admission control policy, which were out of scope for our work. Just to be compliant to PHOENIX's applications, the granted QoS addressed the requirement of a 10 ms delay for the 99% of packets for the reference third (and fourth) queue.

By looking at the results reported in the two tables above, the proportional relative delay differentiation between the

4 queues was more respected with a more rigid filter. However, the optimum value of k depends on the bursty nature of the traffic; we have to set k to a lower value to follow better the rate variations in case of more bursty aggregates.

The aim is to properly balance a responsive enough system dynamics with stability and consistency requirements that in the end lead to a stricter respect of the proportionality of the delays experienced by packets in the different queues, according to the assigned QoS factors.

We also changed the nature of the filter, using a sort of moving average. The filtering process was as follows:

$B(n) = [B(n-1)+B(n-2)+\ldots+B(n-M)] / M$

where B(n) is again the average buffer dimension at the n-th interval and M the number of the intervals taken into account in the averaging calculus.

By Analysing the average delay with M=8 or M=16, we realized how increasing M the filter became more rigid and less receptive to the variation of the rate. The figures for the 99th percentile were: 24.5, 21.3, 11.6, 8.3 ms for M=8 and 23.1, 21, 11.9, 10.1 ms for M=16. Hence, slightly better results were obtained in the first case.

Furthermore, we investigated a different measurement window of the low pass filter; it was set to 80, 160 and 320 ms (the former value was 40 ms). We point out that in the various simulations the optimum value for the k factor was taken (0.86 , 0.8 and 0.69, for the measurement windows 80, 160 and 320 ms, respectively).

Increasing the measurement window of the filter we obtained more averaged rates. This way, the system reactivity goes down; hence the instantaneous variations of the traffic have a less impact. Thus, we needed to set k to a lower value in order to respect as precisely as possible the proportionality between the 99th percentile of the delays experienced by packets in the 4 queues, but we could not achieve the same performance as in the 40 ms case.

The 99th percentiles of the delays were 20.1, 16.8, 11, 7.9 with an 80 ms measurement window and a 0.86 k filter, and 18, 16.1, 11.3, 9.1 with a 160 ms measurement window and a 0.8 k filter, and 18,1 ms, 16.8 ms, 11 ms, 9.4 ms respectively with a 320 ms measurement window and a 0.69 k filter.

Concerning the issue of the proper design and configuration of the traffic measurement process, as last test we considered a measurement system based on thresholds, let us say a threshold of B bytes, the weights updating happens only when at least the buffer dimension of a queue exceeds B.

Certainly, the behaviour of this system strongly depends on the value of B. We made an analysis with B equal to 1 MTU (Maximum Transfer Unit), 2 MTUs or 10 MTUs. The typical value of MTU in an IP network IS 1500 bytes.

The 99th percentile became worse (as usual in terms of consistency with the ratios between the defined QoS parameters) augmenting the value of B.

More precisely, in order of increasing values of B, the issued figures were 24.9 ms, 21.4 ms, 11.7 ms, 8.5 ms; or 24.9, 21.4, 11.4 and 9.2 ms, or 23.2, 20, 12.1 and 11 ms, for the 4 queues respectively. When B was low, the percentile was not too far from the results obtained by a moving average filter with a low value of M and by an optimum low pass filter.

Tables 4 and 5 show the buffer dimensions for the different queues required to avoid packet losses at all (100th perc. Min. buffer size) and to achieve 1 percent of packet loss rate (99th perc. Min. buffer size), in the two cases of 4 and 7 queues to be managed by the dynamic WFQ scheduler. The Pi parameters were set to 1, 2, 3, 4, 1.5, 2.5 and 3.5, respectively for the different queues in the latter case.

Queue	99th perc. Min. buffer size (bytes)	100th perc. Min. buffer size (bytes)
1	38254	58521
2	35212	54789
3	34457	53541
4	32587	51512

Table 4 – 99th and 100th perc. Min. buffer sizes in the different 4 queues

Queue	99th perc. Min. buffer size (bytes)	100th perc. Min. buffer size (bytes)
1	48981	69842
2	45889	67854
3	44491	65305
4	41672	61478
5	48605	69001
6	45302	65991
7	43510	64215

Table 5 – 99th and 100th perc. Min. buffer sizes in the different 7 queues

Queue	Average Delay (ms)	Average Delay at the interface (ms)
1	6.45	
2	5.29	
3	4.15	
4	3.22	4.81

Table 6– Average Delay in each queue and at the interface as a whole, for the 4 queues case

Queue	Average Delay (ms)	Average Delay at the interface (ms)
1	7.12	
2	5.77	
3	4.20	
4	3.31	
5	6.18	
6	4.99	
7	3.64	5.11

Table 7– Average Delay in each queue and at the interface as a whole, for the 7 queues case

The last column reports values about 50% higher than the second. To save 1 percent of the packets we needed to spend a lot in terms of buffer space. As expected, the performance of the system with 7 queues was worse (bigger buffer sizes were required).

In tables 6 and 7 are shown the average delay for each queue and at the interface in general, for the two cases of 4 and 7 queues. The calculus of the Delay essentially encompassed the queuing delay (determined by the specifically deployed scheduling discipline) and the

transmission delay (a minor contribution, considering the MTU value and the link capacity of the concerned interface).

Although the two investigated systems had the same capacity for queue (12.8 Mbit/s), the latter was a little bit slower. This is due to the higher complexity of the system in presence of more queues that leads to a smaller efficiency.

C. *Design Consideration*

The choice of each mechanism and parameter concerned in the scheduling system is important to obtain the desired performance.

Before considering the different factors we have to remember that the WFQ is not an ideal scheduling scheme. Therefore, the link required to obtain the same performance as in the corresponding GPS ideal case must be higher. Only after a proper dimensioning of it, we can focus on the design options.

First, it is fundamental to set the window filter in relation to the nature of the traffic. As expected, we have realized that the shorter the measurement window, the better; because in this case the reactivity is higher and we can follow well the traffic rate variations.

A second step is to choose the best filter. We made experiments with three different filters: low pass filter, thresholds based filter and moving average filter. We have demonstrated that the former two are similar in terms of achieved performance if we used a quite small k parameter, anyway close to the unit, or thresholds, of the order of some MTUs, respectively. The bigger the values, the more static the filters. Such a selection is more suitable for traffic with not that high burstiness. A low burstiness could be due either to not too time-variant flows or a high level of multiplexing, which is quite common in a backbone network, or in a Differentiated Services architecture over a wide bandwidth infrastructure. The moving average filter is effective as well, if employed with a small number of considered buffer size samples.

Hence, the decisive aspect is the burstiness nature of the concerned traffic rather than the choice of a specific filtering process, at least from a performance point of view, while from a computational point of view the simpler, the better.

Another important issue is to set the buffer dimension. We have to consider that the gap from 1 to 0 percent of packet loss is high. About 50% more of buffer space was required in fairly common working conditions.

It is important to take into account the number of queues to be managed in relationships of course, to the entering traffic and the interface capacity. The system complexity increases with the number of configured queues, inevitably leading to a less strict control of the guaranteed delays (in terms of consistency with the ratios between the assigned QoS parameters). To be noted that a higher difference between the ratios of the Pi factors allows to better differentiate between the QoS provided to the traffic aggregates of each queue.

At the end, it is not a trivial task to design a scheduling system as efficient as possible, but it is extremely helpful to acquire some a priori information about the actual working conditions.

IV. CONCLUSIONS

In this work, we have investigated and analyzed several possible design options and configuration parameters for a dynamic WFQ scheduler.

We have demonstrated that it is possible to provide proportional relative QoS guarantees consistently to pre-assigned factors to each queue, and exploiting well the available bandwidth of a given interface according to the actual arrival rate of the traffic classes, which are time-variant in nature.

More complex working conditions, such as in the case of a high number of queues, affect the achievable performance in terms of strict control of the granted QoS.

However, even absolute QoS guarantees suitable for real-time multimedia applications, as in the context of the PHOENIX project, can be obtained with a proper resource provisioning and admission control policy.

Finally, guidelines for a designer and a network operator have been carried out on the basis of the performed analysis.

REFERENCES

[1] J. Wroclawski, *The Use of RSVP with IETF Integrated Services*, RFC 2210. IETF intserv WG. September 1997

[2] K. Nichols et Al. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, RFC 2474. IETF difsserv WG. December 1998.

[3] S. Blake et Al, *An Architecture for Differentiated Services*, RFC 2475. IETF diffserv WG. December 1998.

[4] S. Keshav, *An Engineering Approach to Computer Networking*, Addison-Wesley professional computing series, 1997

[5] Chin-Chang Li, Shiao-Li Tsao, Meng Cheng Chen, Yeali Sun, Yueh-Min Huang, *Proportional Delay Differentiation Service Based on Weighted Fair Queuing*

[6] Kun Pang Xiaokang Lin Junli Zheng Xuedao Gu Nat, *Dynamic WFQ scheduling for real-time traffic in wireless ATM links*, Communication Technology Proceedings, 2000

[7] H.263/MPEG4-compressed video traces: http://www-tkn.ee.tu-berlin.de/research/trace/trace.html