I.J. Information Technology and Computer Science, **2017, 2, 11-17**
Published Online February 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2017.02.02

Weighted Priority Queuing: A New Scheduling Strategy for Web Services

Randa. Hammami
ReDCAD Laboratory, Sfax, Tunisia
E-mail: randa.hammami@ieee.org

Yossra. Hadj Kacem, Senda. Souissi, Hatem. Bellaaj and Ahmed. Hadj Kacem
ReDCAD Laboratory, Sfax, Tunisia
E-mail: {hatem.bellaaj@redcad.org, ahmed.hadjkacem@fsegs.rnu.tn}

***Abstract*—**Web services are considered as one of the best and most widespread solution for handling the interoperability problem and the challenge of integration. The proliferation of Web services over the Internet becomes more and more significant. They are henceforth playing an important role in several fields such as e-health, e-commerce and e-learning. Thus, one important question arises: how to manage Web services more efficiently? It is a key problem to the Web services based- applications at present especially that the need to enhance the Quality of Services (QoS) is constantly growing: the better the QoS are, the more the users are satisfied. This has spurred the study of scheduling algorithms for providing QoS guarantees.

In this paper we put the light on the Web services requests scheduling strategies on the server side. In fact, we present a brief overview and a comparative evaluation of three queuing scheduling disciplines for Web services, which are: First in First out (FIFO), Priority Queuing (PQ) and Weighted Fair Queuing (WFQ). Then, we put forward a new scheduling strategy based on two well-known strategies namely: Priority Queuing and Weighted Fair Queuing. The experimental results highlight the merits and shortcomings of each scheduling discipline in addition to its performance in terms of: execution time, communication time and response time.

***Index Terms*—**First in First out, Priority Queuing, Weighted Fair Queuing, QoS, execution time, communication time, response time.

I. Introduction

With the emergence of the Service Oriented Architecture (SOA), business process components are more and more being decoupled. SOA is "a software architecture where functionality is grouped around business processes and packaged as inter operable services. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services" [15].

Web Services (WSs) are one of the most active and widely adopted implementation of SOA. WSs technologies are being continuously standardized to ensure interoperability and security. Using Web services in SOA creates a wide network of services that collaborate in order to implement complex tasks. Reference [1] defines a Web service as a "platform-independent Web application, which is self-contained, self-describing application that can be published, located, called through the Web". Thus, we can affirm that a Web service is a software system that allows applications to interact across the network using the available protocols of transportation. This communication is based on the principle of requests and responses as described by Fig. 1. In fact, architecture for service-based applications has three main parts: a provider, a consumer, and a registry [2]. *Providers* publish or announce their services which provide functions or business operations that can be deployed over the Internet and accessed through the *registry*. *Consumers* find and then invoke them look for a suitable WS by sending some requests to the *Web service registry* then invoke the selected WS.

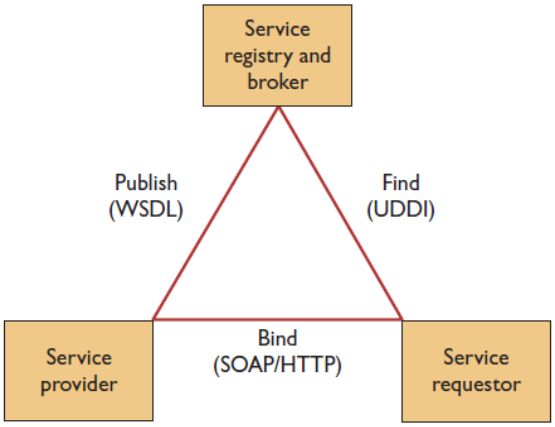

Fig.1. Web services architectural model [2]

Since the proliferation of SOA is leading to the emergence of large sets of WSs in different fields, it becomes vital to monitor the Quality of Services (QoS). Reference [7] refers to QoS as the ability of a service to respond adequately to its consumers' demands which can

be explicitly or implicitly expressed. The service aims to satisfy users. These requirements may be related to several aspects of a service, e.g. availability, reliability etc...

Providing QoS-guaranteed services is necessary to satisfy WSs' users. While a well designed Web server should not be persistently overloaded i.e. work enters the Web server at a greater rate than the Web server can complete it, scheduling incoming requests turns to be crucial. Hence, an efficient scheduling strategy is required to influence the decision of service provider for cost benefit [16].

A scheduler may be regarded as a queuing system consisting of a server providing service to a set of customers. The customers queue packets for service, which are chosen by the scheduler for transmission based on a service discipline defined by the scheduling algorithm. The service discipline must be designed to meet the desired QoS requirements of individual customers [3]. Several scheduling disciplines are well-known in the literature. Hence, this paper deals with a key challenge: Web services scheduling strategies. In this paper, we focus on First In First Out (FIFO), Priority Queuing (PQ) and Weighted Fair Queuing (WFQ) algorithms. Then we suggest a new scheduling policy called Weighted Priority Queuing (WPQ) to decide the order in which requests should be serviced.

The paper has been structured as apart from introduction in section 1, section 2 covers a brief overview of FIFO, PQ and WFQ scheduling strategies. Then, we introduce the proposed WPQ strategy in section 3. Section 4 is an evaluation of the performance of the four queuing disciplines mentioned above. Finally paper has been concluded with section 5.

II. Scheduling Strategies: Overview

FIFO, PQ and WFQ are the basic scheduling strategies and among the most used ones. For this reason, this section details them to a certain level.

A. First In First Out

Historically, First In First Out (FIFO) scheduling was one of the first scheduling policies to be implemented in time-sharing systems. Due to its simplicity, it is widely supported. For example, it is implemented in the Linux kernel to schedule tasks with high priority [8].

In FIFO, all incoming requests from different consumers are placed into the same queue. Then, they are treated in an equal way: the server proceeds to requests one by one in the same order in which they were placed in the queue [9]. In other words, the first request that comes to the queue is the first one to be served as shown in Fig. 2.

FIFO behavior is predictable and needs a low calculation since the request delay depends on the queue size. However, a request can be dropped in case it arrives to a completely full queue no matter how important is the request.

B. Priority Queuing

Priority Queuing is the basis for various queuing mechanisms. Requests are first classified into different priorities. Then it places them into separate queues [10] (see Fig. 3). Each priority class uses one FIFO (First In First Out) queue, where requests are dropped if the queue is full. All requests having the highest priority are served before packets having lower priority.

The low computational load on the system and the set of priorities are the important benefits of this discipline [11]. On the other hand, the large traffic of the high priority requests can lead to the timeout of lower priority requests and so they will be dropped by the server [12]. Thus, the need of a specific policy to condition the high priority traffic is essential. Moreover, a misbehaving high-priority request can increase significantly the delay and jitter experienced by other high-priority requests sharing the same queue.

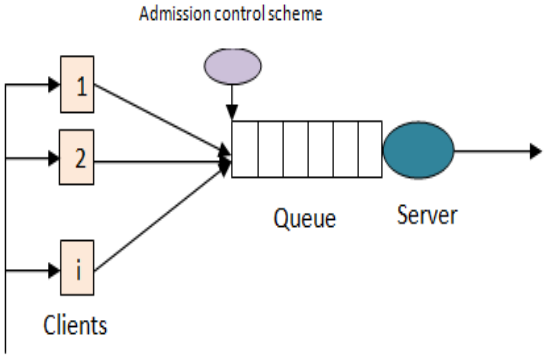

Fig.2. FIFO Queuing mechanism

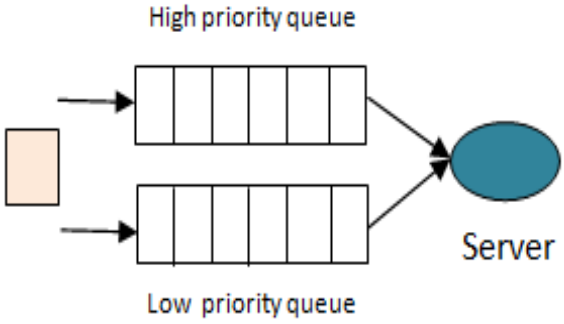

Fig.3. Priority queuing mechanism

C. Weighted Fair Queuing

Weighted Fair Queuing was first introduced in 1989. WFQ supports flow with different bandwidth requirements by giving each queue a weight that assigns it a different rate of the servicing bandwidth [13]. WFQ schedules incoming requests by calculating a virtual finish time based on their arrival time, size and associated weight. The virtual finish time here represents time at which the server would finish to serve the request. Then, WFQ arranges requests in the ascending order of the virtual finish time [4]. This can guarantee that each queue gets its shares of bandwidth in a proportional way to the allocated weights [14]. Fig. 4 shows an example of how the scheduler treats the incoming packets under the WFQ mechanism.

The WFQ algorithm is a little bit complex due to the necessary calculation procedure which requires the server to save a significant amount of data. This can impact the scalability of the WFQ policy when attempting to support a large number of requests' classes on high-speed applications.

We also find out the advantages and disadvantages of each scheduling strategy to help readers to implement a new scheduling discipline or to select the most appropriate existing one. Table 1 summarizes the relative merits and shortcomings of each discussed mechanism.

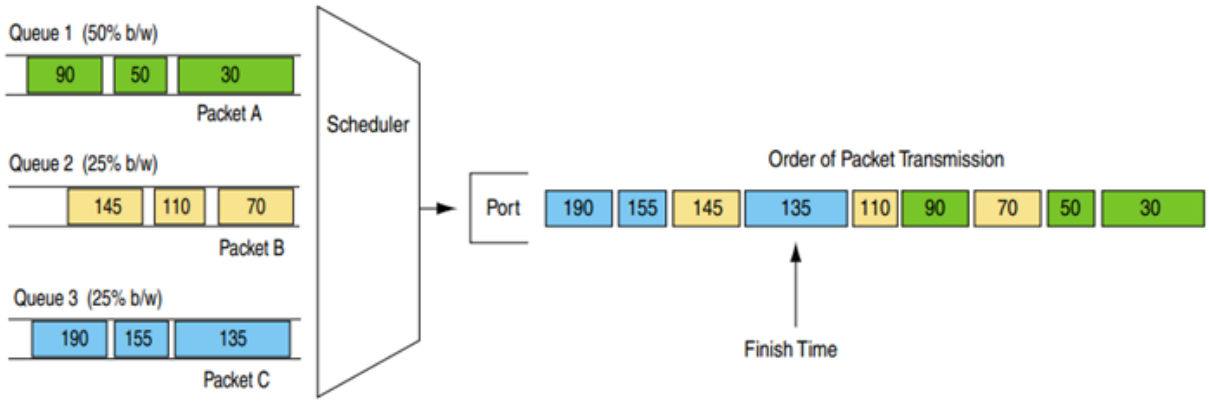

Fig.4. Weighted Fair Queuing mechanism

Table 1. Summary of merits and shortcoming of scheduling strategies

	Merits	Shortcomings
FIFO	- Easy to implement - Low computational load on the server as far as the number of requests in the queue is low	- FIFO queue can cause delays, jitter, which is not very suitable for real-time applications - A request may be dropped if it exceeds its timeout while waiting in the queue
PQ	- Suitable for systems that need to differentiate between requests which are time-critical	- The high priority traffic should be conditioned to decrease lower priority traffic delay - If the high-priority traffic volume becomes excessive, low-priority requests can be dropped
WFQ	- It ensures a fair and balanced sharing of the bandwidth to service requests with a limited delay.	- It implements a complex algorithm which requires the saving of a significant amount of data to be analyzed (e.g. the starting and end times of each request…)

III. Proposed Scheduling Strategy: Weighted Priority Queuing

As we stated above, the PQ scheduling strategy may cause some delay or low priority requests' abandon. In other hand, the WFQ scheduling strategy is more complex due to the calculation of requests' virtual finishing time. So, in order to overcome these limitations, we decided to combine the two scheduling strategies together.

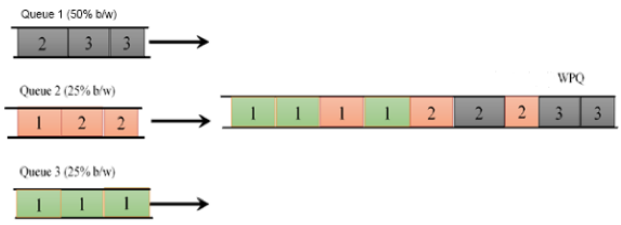

Fig.5. WPQ scheduling strategy example

The main idea of WPQ is to give a priority to incoming requests to order them. And we keep the weights assigned to each queue to balance the traffic. Thus, the WPQ algorithm includes 2 main phases: priority assignment (from priority 1 to priority 3) and weighted queue assignment as described by Fig. 5.

IV. Simulation Results

To evaluate the performance of the new scheduling method along with the three other queuing scheduling disciplines mentioned above, we created a client-server program using the Java programming language. We also developed some restful Web services to be accessed from the server in JSON format. Thus, the server receives some requests (to be placed in queues according to the scheduling strategy), executes them and finally it sends responses back to the clients. We have implemented the PQ, WFQ and WPQ scheduling strategies algorithms and have run simulations to accumulate data and compare these strategies. The simulation programs are executed on a PC machine with 4 GB RAM and Microsoft Windows Operating System.

A. Performance metrics

The experiments focus on the study of the behavior of the implemented scheduling disciplines (PQ, WFQ and WPQ) based on 3 QoS parameters which are: the execution time, the communication time and the response time which are indeed very critical in the evaluation of QoS. These QoS parameters are detailed below (see Fig. 6) [5]:

- *Response time*: is defined as the time elapsed between sending a request and receiving its response; $T_{resp} = t_4 - t_1$.
  It is the make span from user submitting a job until receiving the results. It depends on both the workload and system performance, and the network transmission time depends on the network latency and the input data size [6];
- *Execution time*: is defined as the time elapsed for processing a request; $T_{exec} = t_3 - t_2$;
- *Communication time*: defined as the round trip time of a request and its response; $T_{comm} = T_{resp} - T_{exec}$.

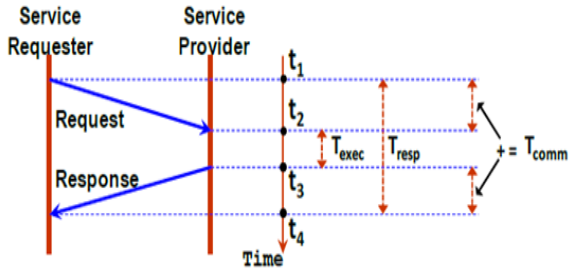

Fig.6. Measured times for QoS monitoring [5]

B. Performance evaluation

Based on our studies, we choose to evaluate the performances of each scheduling policy in two cases:

- The server has to serve 30 clients/ incoming requests concurrently. The rest time of the server is fixed to 10 seconds. This value was selected based on the number of considered objects in the statistical studies' samples.
- The server has to serve 100 clients/ incoming requests concurrently. The rest time of the server is fixed to 50 seconds. So that the queues are completely full.

In all the graphics to be discussed below:

- The horizontal axe represents the clients' number;
- The vertical axe indicates the time in milliseconds.

We compare the FIFO, PQ and WFQ performances then we compare the PQ, WFQ and WPQ performances.

1) Execution time

For 30 incoming requests, the FIFO strategy outperforms largely both of the PQ and WFQ strategies. Afterward, PQ strategy takes less time to execute the incoming requests (Fig. 7).

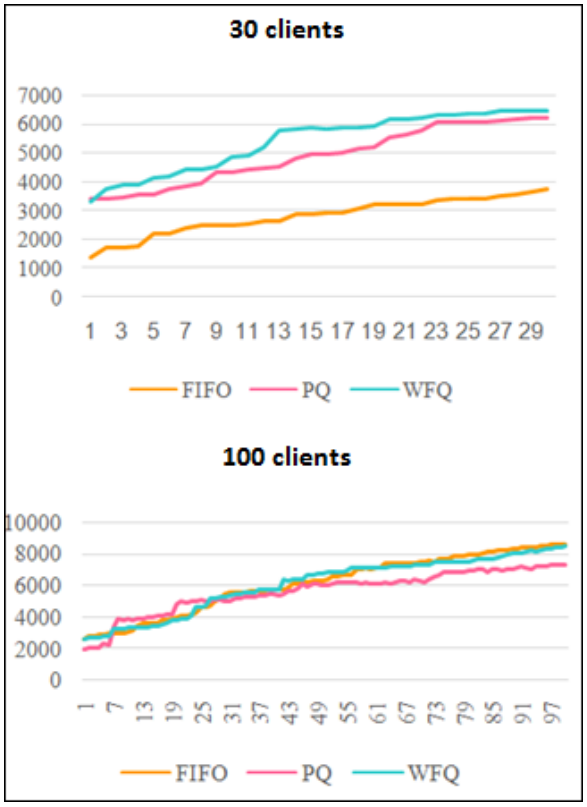

Fig.7. FIFO, PQ and WFQ: execution time

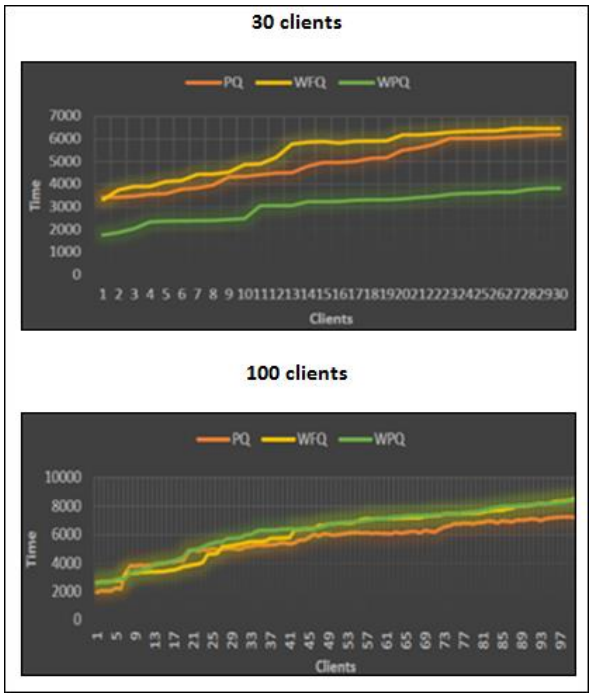

Fig.8. WPQ: execution time

2) Communication time

In both cases, WFQ has the higher communication time as shown in Fig. 9. For a large number of clients, FIFO is by far the better strategy to adopt if one needs to lower the communication time. Besides, it has the most stable behavior in both cases.

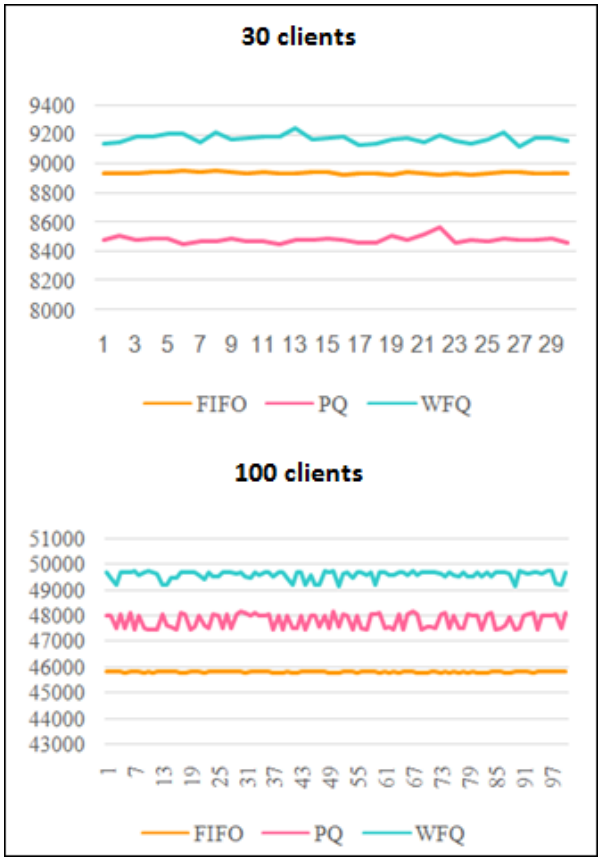

Fig.9. FIFO, PQ and WFQ: communication time

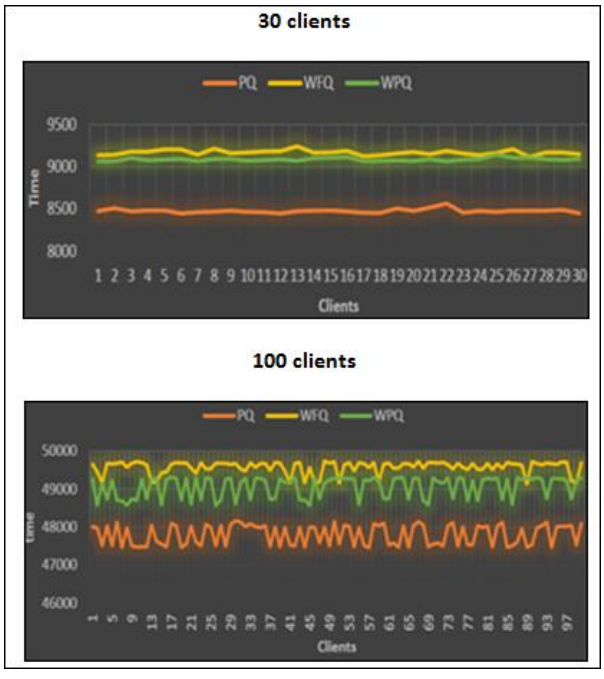

Fig.10. WPQ: communication time

Communication time does not increase linearly with clients as we can see in Fig. 10. This figure summarizes our results showing that in both cases, the PQ policy presents the lowest communication time and the WFQ presents the biggest one. This can be explained by the fact that calculating the virtual finish time of each request increases the time spent by the request on the server while waiting to be serviced. The WPQ is a compromise between the PQ and the WFQ strategies.

3) Response time

Since the response time is the sum of the communication time and the execution time, we can affirm that FIFO has a better performance globally although the fact that there isn't a wide variance between the performance of all the scheduling strategies as described in Fig. 11.

Here again, the WPQ is better than the 2 other strategies when the number of incoming requests is relatively low (30 clients). Then it becomes more and more similar to the WFQ strategy when the number of clients increases to reach 100 ones. In this case, using the PQ strategy guaranties the lowest response time which may satisfy users who need to be served fast (see Fig. 12).

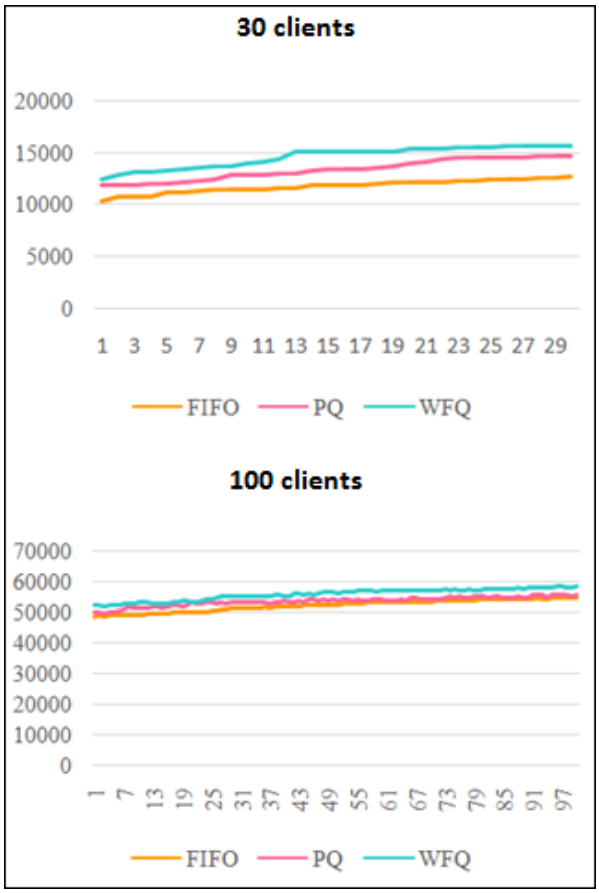

Fig.11. FIFO, PQ and WFQ: response time

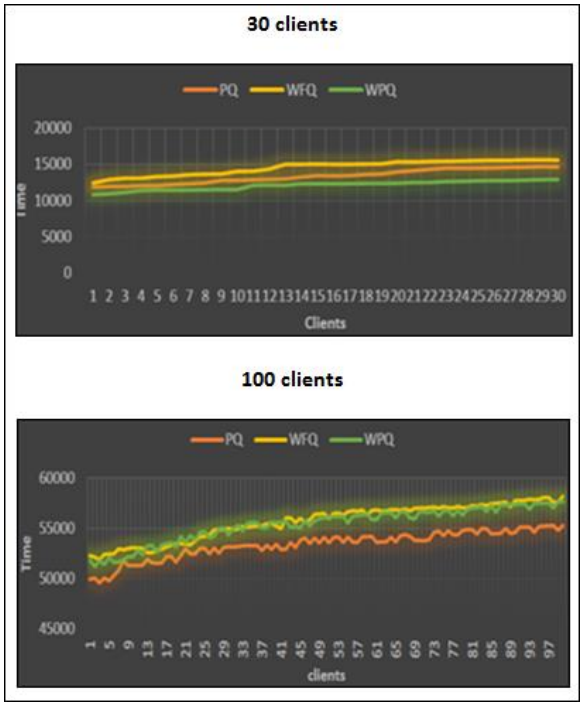

Fig.12. WPQ: response time

4) Summary

Table 2 shows the average values of the execution, response and communication times of the four scheduling methods that we studied. In both experiments dealing with 30 and 100 clients, WFQ is the worst scheduling strategy.

Moreover, we can deduce that the proposed strategy, WPQ is quite well-adapted for a low incoming requests number.

Table 2. Experiments' average values

		FIFO	PQ	WFQ	WPQ
Execution time (ms)	***30 clients***	2801	4891	5409	3031
	100 clients	6164	5584	6089	6308
Communication time (ms)	***30 clients***	8931	8478	9169	9087
	100 clients	45803	47801	49578	49062
Response time (ms)	***30 clients***	11732	13369	14579	12118
	100 clients	51968	53386	55667	55371

V. Conclusion

Nowadays, a lot of Web services are proposed in the market. The rapid growth in demand for high-speed and high quality Web services is creating opportunities and challenges for next-generation WSs-based applications' designs. Each service flow has its own type of QoS requirement. Scheduling strategies play an important role in QoS management. So, appropriate measures must be taken to guarantee a level of QoS that satisfy users.

In this paper we studied some scheduling strategies and evaluated their performances so as to select the most adequate strategy when considering the decision-maker's preferences. Based on the above observations, we can notice that each scheduling strategy has its own strengths and weakness but interestingly enough they show complementary strengths. One alternative may be appropriate in a particular environment but not in others. The variety of alternatives for requests scheduling motivates the need for sound criteria to characterize them. Much work remains to be done. We also made some key observations and pointed out that adapting/ changing the scheduling strategy while running may enhance the server performance.

References

[1] Lu, G., Wang, T., Zhang, G., & Li, S. (2012, June). Semantic web services discovery based on domain ontology. In *World Automation Congress (WAC), 2012* (pp. 1-4). IEEE.

[2] Huhns, M. N., & Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *Internet Computing, IEEE*, *9*(1), 75-81.

[3] Varma, A., & Stiliadis, D. (1997). Hardware implementation of fair queuing algorithms for asynchronous transfer mode networks. *Communications Magazine, IEEE*, *35*(12), 54-68.

[4] Shubhangi, R., & Samir, S. (2013). Comparative analysis of different queuing mechanism in heterogeneous networks. International Journal of Advanced Research in Computer and Communication Engineering, 2(8), 3075-3079.

[5] Halima, R. B., Guennoun, M. K., Drira, K., & Jmaiel, M. (2008). Providing predictive self-healing for web services: a qos monitoring and analysis-based approach. Journal of Information Assurance and Security, 3(3), 175-184.

[6] Zhao, L., Ren, Y., Li, M., & Sakurai, K. (2012). Flexible service selection with user-specific QoS support in service-oriented architecture. Journal of Network and Computer Applications, 35(3), 962-973.

[7] Ben Halima, R. (2009). Conception, implantation et expérimentation d'une architecture en bus pour l'auto-réparation des applications distribuées à base de services Web (Doctoral dissertation, Université de Toulouse, Université Toulouse III-Paul Sabatier).

[8] Bovet, D. P., & Cesati, M. (2005). Understanding the Linux kernel. " O'Reilly Media, Inc.".

[9] Sharmal, R., Sehra, S. S., & Sehra, S. K. (2015). Review of Different Queuing Disciplines in VOIP, Video Conferencing and File Transfer. International Journal of Advanced Research in Computer and Communication Engineering, 4(3), 264-267.

[10] Kumar, A., & Garg, A. K. (2011). Queuing Algorithm Based Quality of Service (Qos) For Scheduling Environment Model in Wimax Network with Opnet Modeler. Global Journal of Researches in Engineering Electronic and Electronics Engineering, 11(8).

[11] Mustafa, M. E. (2015). The Effect of Queuing Mechanisms First in First out (FIFO), Priority Queuing (PQ) and Weighted Fair Queuing (WFQ) on Network's Routers and Applications. IJEIR, 4(1), 188-191.

[12] Zoric, S., Kos, M. & Bajric, H.(2012). Comparative Analysis of Scheduling Algorithms for UMTS Traffic in Case of DiffServ Network Congestion.The Fifth International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ), 2012.

[13] Nandhini, S. (2015). Low Latency Weighted Fair

Queuing for Real time flows with Differential Packet Dropping. Indian Journal of Science and Technology, 8(22).

[14] Raghupathikumar, D. & Bommanna Raja, K. (2014). A Combined Low Latency and Weighted Fair Queuing Based Scheduling of an Input-Queued Switch. Journal of Computer Science, 10(8), 1447-1457.

[15] Kumar, P. (2016). Some Observations on Dependency Analysis of SOA Based Systems. International Journal of Information Technology and Computer Science (IJITCS), 8(1), 54.

[16] Kaur, S., & Verma, A. (2012). An efficient approach to genetic algorithm for task scheduling in cloud computing environment. International Journal of Information Technology and Computer Science (IJITCS), 4(10), 74.

Authors' Profiles

Randa. Hammami was born in Sfax in Tunisia. She recently earned a bachelor then a master degree in computer science from the National Engineering School of Sfax. She is currently a PhD Student in the same field.

Her research studies focus on interoperability of medical information systems including scheduling strategies for web services.

Yossra. Hadj Kacem has an M.S. degree in computer science. She received her M.S. degree from the Higher Istitute of Computer Science and Multimedia of Sfax, Tunisia in 2016. Her research work focuses on Web services.

Senda. Souissi has an M.S. degree in computer science. She received her M.S. degree from the Higher Istitute of Computer Science and Multimedia of Sfax, Tunisia in 2016. Her research work focuses on Web services.

Hatem. Bellaaj has a PhD degree in computer science and is a member of the ReDCAD Laboratory. He is currently serving as an associate Professor at the preparatory institute for engineering schools of Sfax in Tunisia.

Ahmed. Hadj Kacem is a professor of computer science. He is currently serving as the dean of the Faculty of Economics and Management of Sfax -University of Sfax in Tunisia.

How to cite this paper: Randa. Hammami, Yossra. Hadj Kacem, Senda. Souissi, Hatem. Bellaaj, Ahmed. Hadj Kacem ,"Weighted Priority Queuing: A New Scheduling Strategy for Web Services", International Journal of Information Technology and Computer Science(IJITCS), Vol.9, No.2, pp.11-17, 2017. DOI: 10.5815/ijitcs.2017.02.02