Reactivity-based Quality of Service Strategies for Web Applications

Leonardo Silva   Adriano Pereira   Wagner Meira Jr.

Federal University of Minas Gerais - e-SPEED Lab.
Av. Antônio Carlos 6627 - ICEx - CEP 31270-010
Belo Horizonte – Minas Gerais – Brazil
{leosilva, adrianoc, meira}@dcc.ufmg.br

Abstract

The great success of the Internet has raised new challenges in terms of applications and user satisfaction. Web applications demand requirements, such as performance and scalability, in order to guarantee quality of service (QoS) to users. Due to these requirements, QoS has become a special topic of interest and many mechanisms to provide it have been proposed. Those mechanisms fail to consider aspects related to reactivity, i.e., how the users react to variable server response time. This work addresses the use of reactivity to provide new strategies. We design and evaluate a reactivity-based scheduling mechanism that gives priority according to user behavior. We also propose a hybrid admission control and scheduling mechanism that combines both reactive approaches. The results show benefits in terms of response time and user satisfaction.

1 Introduction

The great success of Web has raised new challenges in terms of applications and user satisfaction. For this reason, basic requirements, such as performance and scalability, became essential to provide Quality of Service (QoS).

Traditional QoS mechanisms, such as admission control and scheduling, fail to consider that users have different characteristics. For example, common admission control policies consider only the load of the server, ignoring the user behavior. Similarly, scheduling policies generally adopt a simple First-In First-Out (FIFO) approach, without considering that users have different levels of tolerance to the Internet QoS provided [3].

Previous works have modeled the user reactivity, i.e., the way users react to the service provided [14], and quantified its impact on the performance of Internet services [17]. Their conclusions raise the question on how to improve the traditional QoS mechanisms with some user-reaction-based knowledge. [16] shows the new admission control approach based on the reactivity that is effective for improving the response time, but may cause an increase in the user impatience due to requests rejection.

In order to improve QoS, in this work we design and evaluate a reactivity-based scheduling mechanism that gives priority according to an user behavior. We also propose a hybrid admission control and scheduling mechanism that combines both reactive approaches.

The paper is organized as follows. Section 2 presents an overview of related work. Section 3 explains the concept of reactivity and how it can be modeled. Section 4 describes our new admission control and scheduling approaches. Section 5 discusses the experimental study. Finally, Section 6 presents our conclusions.

2 Related Work

In order to improve QoS we intend to use information about user reactivity, i.e., the user behavior in response to variable performance, to design strategies more suitable for real scenarios. User behavior has been analyzed through several criteria, such as the nature of submitted requests, types of services [11], among other information.

Improving QoS has been a challenging task addressed by many works in diverse research areas. QoS for Internet servers has evolved significantly and most of the works in this area proposes architectures and strategies, such as admission control, load balancing, resource allocation, and scheduling. [6] develops a session-based admission control that rejects new customers during overload periods. [2] proposes a mechanism to adapt the content provided instead of rejecting requests to decrease the server load. [4] discusses classifying web requests into three levels of priority using information as IP address and requested web sites.

Many scheduling and admission control mechanisms have been proposed. [7] investigates the effect of aggregation on the performance using Priority Queuing (PQ) and Weighted Fair Queueing (WFQ) scheduling algorithms. [5] exploits the dependence among session-based requests and propose a scheduling algorithm to control overloads in web servers. [18] uses only one queue for all web requests with
an advanced scheduling rule to differentiate services. Nevertheless, these solutions fail to provide a robust solution for the Internet QoS. We believe that the use of user-reaction-based information is an effective way to design strategies more suitable to user demands, improving Internet QoS. In order to fill this gap, [14] presents a methodology to characterize, understand, and model the user behavior, based on their interaction with servers and their perceived performance. As we explain on Section 3, this model is used as the starting point to our work.

3 Reactivity in Internet Services

Reactivity represents the way a user behaves according to the quality of service provided. As the QoS provided by the server changes, the user behaves differently. [14] presents a characterization model named USAR that makes possible to model the reactivity.

USAR models the reactivity using the following function to relate the inter-arrival time (IAT) and the response time (the time a service takes to process a request, considering its receipt, processing and response):

\[ RAT(k) = \begin{cases} 
I(k,k+1)/R(k), & \text{DIF}(k) > 0 \\
R(k)/I(k,k+1), & \text{DIF}(k) < 0 \text{ and} \\
1, & \text{DIF}(k) = 0
\end{cases} \]

\[ DIF(k) = I(k,k+1) - R(k), \forall k \in \text{workload}, \]

where \( k \) is a user request, \( I(k,k+1) \) is the IAT between request \( k \) and \( k+1 \), and \( R(k) \) is the response time associated to the request \( k \).

Functions RAT and DIF may be used in the discretization model depicted in Figure 1. The \( x \) axis is associated with the DIF function and the \( y \) axis with the RAT function. The model defines seven user reaction classes (A to G), using six limit values in our case study. Values \( k_1 \) and \( k_2 \) divide the positive and negative sides of DIF function, defining a zone close to zero, where the values of IAT and response time are very close to each other. Values \( k_3 \) and \( k_4 \) divide the vertical scale into three different zones, according to RAT function that quantify the correlation between IAT and response time. The classes A, B and C represent behaviors where users do not wait for the answer to their requests before asking another object, and the classes E, F and G represent behaviors where users wait for the answer to their requests before asking another one. The boundaries of these classes are defined by two other constants: \( k_5 \) and \( k_6 \). For Class D, the user requests a new object a short time after receiving the previous one.

Figure 2 presents the patience scale formed by the classes derived from the discretization model. The left side of the scale represents impatient user reaction classes. The right side represents patient user reaction classes.

Besides this model, [14] also shows how to replicate the user reaction classes based on the results of the characterization phase. It means that it is possible to replicate the reactivity, generating a synthetic log that mimics the behavior of users.

The impact of reactive workloads on the performance of Internet services is discussed in [17]. That work uses the USAR model to evaluate the performance of a Web application. It presents a new version httperf [13] capable of reproducing the user reactivity. The experiments demonstrate that different response times affect the workload, changing the rate of requests submitted and, consequently, changing the server load. Therefore, it is shown that server-side affects the client-side behavior, and vice-versa, proving the importance of the reactivity.

The main consequence of demonstrating that reactivity impacts the server performance is that the actual models of Web performance may be improved since they do not consider it. This motivates the investigation of new QoS strategies that may even mitigate negative effects of the reactivity and reinforce the positive ones. Section 4 proposes innovative mechanisms for improving QoS through reactivity.

4 Reactivity-based QoS Strategies

In this section we present new QoS strategies that consider reactivity. First it is important to introduce the concept of burst, since our policies are based on it. Bursts consist of sequences of requests for fetching a web page and its embedded objects (like pictures). A burst is submitted to the server when a user clicks on a link or requests a Web page during its session. Bursts mimic the typical browser behavior where a click causes the browser to first request the selected Web object and then its embedded objects. A session consists of a sequence of bursts in which the time between any two consecutive bursts is below a certain threshold.

The proposed strategies are based on admission control and scheduling strategies. The basic idea is that bursts must be classified into user reaction classes, based on the reactivity model described in Section 3 that establishes seven user classes.
classes from A to G. The main novelty of this paper is the use of the reactivity to improve QoS and the idea to combine the proposed admission control and scheduling techniques.

4.1 Admission Control

Admission control rejects requests whenever the arrival rate is too high and it is necessary to maintain an acceptable load in the system. Without admission control, the response time increases when the system saturates. Traditionally, server utilization or queue length are criteria used in admission control schemes. This section presents briefly the approaches to implement the reactive admission control presented in [16].

4.1.1 Burst-Based Approach

Based on the USAR model, this approach considers how users tend to react according to Internet service’s performance. The burst-based policy rejects bursts when it identifies a fulfilled rejection rule. Traditional policies adopt just one limit of response time and begin to reject bursts once this limit is achieved. We define the policy as a function of the response time of a service ($R$) according to the following rules:

- $\alpha \leq R < \beta$: reject bursts of classes $A$, $B$ and $C$;
- $\beta \leq R < \theta$: reject bursts of classes $A$, $B$, $C$ and $D$;
- $R \geq \theta$: reject bursts of all user action classes.

In these rules, $\alpha$, $\beta$ and $\theta$ are values determined based on empirical results and literature criteria [10].

The idea of this policy is that users who have more impatient profile tend to react faster (that is, reload or submit a different request) than other users when the server presents high response times, degrading server’s performance. The burst-based policy has a multiple criteria rule, minimizing the rejection impact, once less users may have bursts refused by the QoS admission control policy. In summary, this policy has the premise that, under overload scenarios, it may be better to give priority to users who have more chance to wait for the response.

4.1.2 Session-Based Approach

The session-based admission control policy rejects user sessions. Traditional policies employ a single response time threshold and start to reject all user sessions once this limit is reached. The burst-based admission control policy may affect all users, but the session-based policy is different, since it tends to affect fewer users. Considering this, reactivity may be important to identify which user sessions have to be dropped.

We monitor the average response time ($R$) and the user session profile for each session ($USP$), i.e., the average user reaction class of each burst that has already been served. We define the following three criteria:

- $\alpha \leq R < \beta$: reject user sessions with $USP < 4$, i.e., sessions associated with classes $A$, $B$ and $C$;
- $\beta \leq R < \theta$: reject user sessions with $USP < 5$, i.e., sessions associated with classes $A$, $B$, $C$, and $D$;
- $R \geq \theta$: reject all user sessions.

This is based on the same idea presented in the reactive burst-based policy, however applied to session-based control mechanism. The next subsection describes scheduling.

4.2 Scheduling

Most existing web servers provide services based on the Best Effort approach, using FIFO scheduling [18]. As presented in Section 2, preview works proposed different approaches that present gains, but fails to consider the dynamics of the reactivity.

4.2.1 PFIN Scheduling Approach

In Patient-First Impatient-Next (PFIN) approach, bursts classified with impatient classes ($A$, $B$, and $C$) go to the low priority queues, and those with patient classes go to the high priority ones.

We propose two variations of the PFIN approach in terms of the number of priority queues used to schedule the bursts. In Figure 3 there are three queues. All the bursts identified with impatient classes are scheduled to the low priority queue and those identified with patient classes are scheduled to the high priority one. Bursts identified with classes $A$, $B$ or $C$ go to queue 0, bursts classified with $D$ go to queue 1, and finally, the ones classified with classes $E$, $F$ and $G$ go to queue 2. In the other variation we employ seven queues, one for each user class. The most patient classes get the high priority queues.

The PFIN policy is based on the idea that when the load is increasing the users who have more patient profiles tend to present a lower load to the server than the impatient ones since after receiving its response, they take more time to proceed and submit another request. Users with impatient
profiles tend to react faster, asking requests even before receiving the previous one. This policy has the premise that under overload scenarios it may be better to give priority to users who have more chance to spend more time to submit requests, slowing down the server load and increasing the user satisfaction.

4.2.2 IFPN Scheduling Approach

In Impatient-First Patient-Next (IFPN) approach, requests classified with impatient classes (A, B, and C) go to the high priority queues, and those with patient classes (E, F, and G) go to the low ones. Like PFIN approach, two variations are proposed with three or seven priority queues.

The IFPN policy is based on the idea that when the load is increasing it is better to answer first the impatient users, in order to increase their satisfaction. The advantage of delaying answers to patient instead of impatient users is the fact that the satisfaction of patient users tends to take more time to degrade. In summary, this policy has the premise that it may be better to give less priority to users that have more chance to wait for the response to their requests.

4.3 Admission Control and Scheduling

We propose a hybrid three-level approach, that combines the two admission control strategies and scheduling. The idea of this new approach is to put together the advantages of each one. Admission control is good to avoid raising the response time to unacceptable values [16]. Scheduling is adopted to control the burst’s priority according to the user class, providing a reduction in the burst’s expiration rate, measure related to user satisfaction as we may observe on Section 5.

In the session-based approach, rejection of sessions is drastically started as response time grows. In the two-level approach, first of all, burst rejection is started, before the rejection of sessions. This strategy smooths the session rejection through a previous step. Once the burst rejection is not effective to slow down the response time, session rejection is activated. In Parallel, scheduling produces a burst reordering process that can reduce the number of unsatisfied users. We define the following criteria:

- $\alpha_1 \leq R < \beta_1$: reject bursts of classes A, B and C;
- $\beta_1 \leq R < \theta_1$: reject bursts of classes A, B, C and D;
- $R \geq \theta_1$: reject bursts of all user action classes;
- $\alpha_2 \leq R < \beta_2$: reject user sessions with USP < 4, i.e., with average user reaction classes A, B or C;
- $\beta_2 \leq R < \theta_2$: reject user sessions with USP < 5, i.e., with average user reaction classes A, B, C, or D;
- $R \geq \theta_2$: reject all user sessions;
- $R > \gamma$: turn on the scheduling policy.

This strategy rejects both bursts and sessions, but according to different limit values, balancing the burst’s rejection and providing a way to schedule the bursts in order to raise the user satisfaction.

5 Experimental Study

This section presents our experimental analysis. Section 5.1 briefly describes the simulator used in our experiments. Section 5.2 explains the methodology to evaluate the new QoS policies. Section 5.3 shows the experiments and results. Finally, Section 5.4 presents a discussion about the experimental evaluation.

5.1 Simulating Reactivity and QoS Policies

In order to evaluate the new scheduling strategies we extended the USAR-QoS simulator presented in [16] with novel features, capable of reproducing the behavior of different scheduling mechanisms. The architecture of the simulator is based on a real web application composed of a server providing a certain service and a set of clients. The server is composed of one or more queues and a processing unit with a certain limited processing capacity. The clients behavior is based on the reactive version of the httpperf workload generator presented in [17].

The USAR-QoS’ architecture is based on events and respects modularity. It uses the Simpack Toolkit [8]. More details about it can be found in [15].

The USAR-QoS is prepared to record the following information about each simulation:

- Bursts Throughput: the rate of bursts answered (replied) by the server, and requested by users (requested) at each period of time.
- Bursts Expired Throughput: the rate of bursts that the users request the next one before receiving its response, due to impatience and high response times.
- Response time: user perceived response time, comprising the time interval between the request of a burst and the time the client finishes to receive the response.
- Server Queue Size: number of bursts waiting to be served in the server queues.
- Server Utilization: proportion of time the server is busy.
- Number of Active Sessions: number of sessions active, i.e., sessions that have not finished.
- Cumulative number of sessions: cumulative number of completed sessions.
Information of throughput and response time are recorded also applying a smoothing function named smooth bezier that provides a better observation of the overall behavior of data.

5.2 Methodology

In order to evaluate the effectiveness of the proposed reactive QoS policies, we simulate them using the USAR-QoS simulator. We prepare a TPC-W-based synthetic workload trace file to be used as the input for the simulation.

We simulate several scenarios using different USAR-QoS configurations to observe how the application server behaves. We discuss in this paper a scenario in which the server achieves a high throughput and the response times observed raises over the user satisfaction threshold. We base this threshold on [10] where the authors identify three groups regarding the response time of a system:

- 0.1 sec: the limit when a user perceives that the system is reacting instantaneously.
- 1.0 sec: the limit when the flow of thought of a user is not interrupted, although the user may notice the delay.
- 10.0 sec: the limit when a user loses attention and the interaction with the system is disrupted.

Based on these values we implement each of the proposed reactive admission control and scheduling strategies in the USAR-QoS. As explained in Section 4, each reactive QoS policy has a set of values ($\alpha$, $\beta$, $\theta$ and $\gamma$) that define its functioning. These values should be carefully chosen since the effectiveness of each policy depends on them. The values we choose are: $\alpha_1 = 3.0$, $\beta_1 = 5.0$, $\theta_1 = 7.0$, $\alpha_2 = 5.0$, $\beta_2 = 7.0$, $\theta_2 = 9.0$, and $\gamma = 0.0$.

We also evaluate the basic non-reactive QoS strategies on USAR-QoS. We implement a traditional session and burst admission control mechanisms with a threshold of 9.0 and 7.0 seconds, respectively. We implement also the Best Effort FIFO scheduling approach.

The experiments we present here consist of 5000 sessions, created in an average rate of 10 sessions per second. The server is configured to support 50 bursts per second of throughput. The trace file is based on the TPC-W benchmark [1, 12]. Each burst is identified with a different user reaction class according to the USAR model [14]. The overall distribution of action classes for the trace file is showed by Table 1.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>10%</td>
<td>10%</td>
<td>15%</td>
<td>15%</td>
<td>15%</td>
<td>30%</td>
</tr>
</tbody>
</table>

We evaluate all combinations of reactive and non-reactive admission control and scheduling policies to compare the results and verify the most effective ones. Due to space constraints we show only the most relevant results.

5.3 Results

Table 2 presents the experimental simulation results summary. Each experiment is identified by a number from 1 to 5. The columns admission control and scheduling describes the experiment configuration in terms of the QoS strategies. The table presents the following information: session and burst admission control policy, scheduling policy, total experimental duration, higher response time value (represented as $R$), mean response time, total number of requests, responses and expirations of bursts, burst lost rate, and number of rejected sessions by the session admission control. The burst lost rate represents the percentage of bursts expired compared to the whole number of bursts requested to the server. It corresponds to the number of occurrences the user asks the next burst before receiving the previous response.

In experiment 1 no admission control policy is active and the scheduling performed follows the typical FIFO approach. Figure 4 presents the average response time (a) and the bursts throughput (b). As we observe, the mean response time achieves a value higher than the 10-seconds limit due to the great number of requests being scheduled to the server. The throughput of the server is close to the server limit and there is a great number of bursts that expires during the execution, showing the overload situation. The total simulation time for the execution of the 5,000 sessions is 13,284 seconds. We observe that the burst lost rate achieves 36.98%, representing probably a high number of user unsatisfied.

![Figure 4: Experiment 1: no admission control and FIFO scheduling](image)

(a) Average Response Time  (b) Bursts Throughputs

The simulation performed in experiment 2 uses the reactive admission control policies for session and burst. The scheduling approach is set to run the typical FIFO approach. The response time achieves lower values than experiment 1, showing the experiment configuration is effective to guarantee a better QoS level. Table 2 shows the maximum response time of 7.22 seconds, almost the lower value compared to the other experiments. This is due to the rejection of sessions and bursts by the admission control mechanism. Figure 5 shows the response time achieved (a) and the cumulative throughput (b). It is important to notice that there is a significant difference between the cumulative bursts requested throughput and the cumulative number of bursts.
replied, as we can see in (b). Their difference corresponds to the number of bursts rejected by the reactive mechanism of admission control. Since a great number of bursts and sessions are rejected the load under the server decreases and the response time achieves lower values. Despite the low response times, the user satisfaction is not fulfilled and the burst lost rate achieves 38.62%, showing that a scheduling mechanism may improve the results.

Figure 5: Reactive admission control and FIFO scheduling

In experiment 3 and 4 there is no admission control active but the scheduling mechanism is set to perform the PFN an IFPN approaches, respectively. For experiment 3 the bursts lost rate achieves 25.35%, a lower value than experiments 1 and 2, but the response time achieves 84.07 seconds showing that the scheduling may cause high delays. For experiment 4, the response time achieves 44.37 seconds, but the lost rate is very low (2.55%) showing that the IFPN scheduling is very effective to provide a response time according to the user tolerance to QoS. Worths to remember that such mechanism gives higher priority to requests of impatient users. Those requests are answered first by the server and the impatience level decreases, impacting the burst lost rate.

In order to improve the results, experiment 5 evaluates the hybrid strategy. In this experiment the reactive mechanism of admission control for session and burst and the reactive IFPN scheduling approach are active. As we observe, the maximum response time values observed on Table 2 present the lower value compared to the other experiments. The experiment presents the second lower value for the burst lost rate (21.67%). Figure 6 presents the response time (a) and cumulative throughput (b). We observe the low response time values and a significant percentage of bursts rejected that impacts the burst lost rate value.

5.4 Discussion

As we may observe, the experiments running the reactive strategies achieve better response times and burst lost rates compared to the experiment without any additional mechanism. The experiment 4, running just the IFPN reactive scheduling, achieves the best burst lost rate, otherwise the response time behavior is not the better one. Experiments 2 achieve better response times due to the effectiveness of the rejection of bursts and sessions, despite their high burst lost rate. Experiment 5, running the reactive admission control and scheduling, presents the best result, achieving an equilibrium in terms of response time and burst lost rate values.

Considering these results, we conclude that the reactive QoS approaches present significant improvements, despite the gains are different according to each experiment configuration. Moreover, it is a task of the systems engineer to choose the best approach in order to provide each application demands.

6 Conclusions

This paper proposes and evaluates scheduling and hybrid techniques based on the user reactivity to guarantee QoS in Internet services. We have designed and implemented the USAR-QoS simulator which allows the evaluation of the new QoS strategies considering the dynamic interaction between client and server-sides. We proposed the PFN (patient-first impatient-next) and IFPN (impatient-first patient-next) scheduling approaches. The paper presents also an hybrid approach that combines admission control and scheduling in a single solution designed...
to optimize the benefits of each strategy.

Each approach is evaluated using the USAR-QoS simulator running 5,000 TPC-W-based sessions. We evaluate the new policies comparing the results with the Best Effort FIFO approach scenario with no admission control.

From the results we observe different gains according to each QoS approach. The mechanism based only on reactive scheduling achieves the best burst lost rate, since its mechanism gives priority to requests classified with impatient classes. The mechanisms that adopt the admission control are effective to reduce the response time, but may cause the increase in the burst lost rate, due to the increase in the amount of users rejecting bursts or sessions. The hybrid mechanism presents an equilibrium, reducing both the response time values and the burst lost rate.

It is important to observe that the proposed policies may be unfair to certain users since their typical behavior is used to improve QoS. The benefits obtained are significative and we recommend our policies despite the subject of unfairness.

There is a relevant improvement in the QoS of reactive Internet systems through the use of reactive approaches. As part of ongoing work we plan to study the problem of starvation due to the scheduling, using some mechanism such as fair queuing [9]. We also intend to implement the strategies in a real web server.

References