Computer Networks 53 (2009) 668–679

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

# Analysis of burstiness monitoring and detection in an adaptive Web system

Katja Gilly [a,*], Salvador Alcaraz [a], Carlos Juiz [b], Ramon Puigjaner [b]

[a] Miguel Hernández University, Departamento de Física y Arquitectura de Computadores, Avda. de la Universidad, 03202 Elche, Spain
[b] University of Balearic Islands, Departament de Ciències Matemàtiques i Informàtica, Carretera de Valldemossa, km 7.5, 07071 Palma de Mallorca, Spain

ARTICLE INFO

*Article history:*
Received 19 June 2008
Received in revised form 24 October 2008
Accepted 28 October 2008
Available online 21 November 2008

Responsible Editor: J. Neuman de Souza

*Keywords:*
Internet
Network monitoring
Performance measures
Simulation

ABSTRACT

Due to the heavy tailed pattern of Internet traffic, it is crucial to monitor the incoming arrival rate in a Web system to preserve its performance. In this study, we focus on the arrival rate processing mechanism as part of the design of an adaptive load balancing Web algorithm. The arrival rate is one of the most important metrics to be monitored in a Web site to avoid the possible congestion of Web servers. Six methods are analysed to detect the burstiness of incoming traffic in a Web system. We define six burstiness factors to be individually included in an adaptive load balancing algorithm, which also needs to monitor some Web servers' parameters continuously, such as the arrival rate at the server or its CPU utilization in order to avoid an unexpected overload situation.

We also define adaptive time slot scheduling based on the burstiness factor, which reduces considerably the overhead of the monitoring process by increasing the monitoring frequency when bursty traffic arrives at the system and by decreasing the frequency when no bursts are detected in the arrival rate. Simulation results of the behaviour of the six burstiness factors and adaptive time slot scheduling when sudden changes are detected in the arrival rate are presented and discussed. We have considered a scenario made up of a locally distributed cluster-based Web information system for simulations.



## 1. Introduction

The fact that Internet traffic flows exhibit heavy tailed probability distributions has been widely discussed in the Internet literature [1,2]. As Web traffic inter-arrival times normally follow a heavy tailed distribution, maintaining a good performance of the Web system is normally more complicated than if this distribution were easily predictable. Hence, it is possible that in a few seconds a Web server that is not overloaded may receive an increase in the number of connections, which produces a situation of congestion [3,4]. This occurs when the server reaches the connection number limit it can handle. Even, without reaching this limit, if the client's request for a connection is accepted, the response time for that request may experience a long delay because of the long queue of requests waiting to be served by the Web system [4]. Internet service providers often offer different Quality of Service (QoS) levels to provide different priority to different users. When the congestion situation is severe, admission policies are normally applied. This leads to the challenge of satisfying the performance requirements for different types of requests at all times.

Our main concern in the design of a load balancing Web system is how to monitor some Web servers' parameters in a very adaptive way in order to reduce the algorithm overhead. Some of the Web servers' parameters likely to be monitored are the arrival rate, the CPU/disk utilization, I/O performance, etc. The performance of the nodes that compound the Web system have to be monitored continuously in order to know their status and make the appropriate decisions in case of overload to avoid a possible congestion situation. This can be done in several ways: (i) each time a request arrives at the front-end of the

* Corresponding author. Address: Department of Physics and Computer Architecture, Miguel Hernández University, Avda. de la Universidad, s/n, 03202 Elche (Alicante) Spain. Tel.: +34 966658565; fax: +34 966658814.
*E-mail addresses:* katya@umh.es (K. Gilly), salcaraz@umh.es (S. Alcaraz), cjuiz@uib.es (C. Juiz), putxi@uib.es (R. Puigjaner).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.10.026

Web system; (ii) at fixed times by using static time slot scheduling; or (iii) at non-fixed times by using dynamic time slot scheduling. The overhead introduced by option (i) is the biggest because each time a request arrives at the Web system, Web node parameters are monitored. While option (ii) introduces a constant overhead, option (iii) monitors the system at non-fixed intervals, hence, its overhead will depend on the frequency of those intervals. The drawback of defining monitoring in a constant duration interval schedule (option (ii)) is the choice of monitoring time interval. It is very difficult to set a duration interval that fits with all possible Internet arrival rates at the Web system due to its heavy tailed pattern.

We propose using adaptive time slot scheduling (option (iii)) where the frequency of monitoring depends on a burstiness factor that will increase its value when bursty arrivals reach the system, and decrease it, if no burstiness is detected. The adaptive time interval we define depends directly on this burstiness factor. Therefore, the monitoring task's overhead is related to the burstiness of the arrivals in the Web system and time slot scheduling is completely adaptive to the burstiness detected in the arrival rate that reaches the system.

We have included six burstiness factors in a content-aware load balancing model designed with OPNET Modeler [5] to compare the effect of including different burstiness factors in the Web system performance. A previous study of burstiness in a Web system was presented in a conference paper [6]. The load balancing algorithm used is beyond the scope of this article and is fully described in [7].

The following sections of this paper are organised as follows: Section 2 describes related studies on burstiness analysis in Internet traffic. Section 3 details the definition of monitoring slots. The burstiness factors we have considered for our experiments are detailed in Section 4 and the adaptive time slot scheduling mechanism is described in Section 5. Section 6 details the simulation scenario and shows the results obtained. Finally, we discuss some concluding points and the open problems.

## 2. Related work

In this section, burstiness modelling and detection related research is introduced.

### 2.1. Burstiness detection based on traffic

The pioneers in modelling burstiness are Wang et al. [8]. They analyse the relation between jitter and burstiness in real-time communications. In this paper, the burstiness detection mechanism is defined for individual packets. The authors define the burstiness of the *m*th packet as a measure of time that expresses the distance between the actual arrival time and the right edge of the *m*th packet arrival interval because they consider the servers usually process packets one by one at a constant rate. An implementation of this mechanism has been defined in our simulation scenario. More details and results are described in the following sections.

Burstiness detection based on the traffic rate and independent of individual packets is defined in other papers [9–12]. Menascé and Almeida [9] are the first to introduce a burstiness factor. They define it as the fraction of time during the time slot arrival rate that exceeds the average arrival. In this case, burstiness monitoring is carried out following fixed time slot scheduling. In Section 4, we describe in detail how this burstiness factor is defined and works and introduce some modifications to it. Baryshnikov et al. [10] study how traffic predictions can be very useful to reduce latencies and performance degradation in Web servers. They use linear extrapolation as a prediction technique and state that this technique for burstiness detection is not a good predictor. In general, however, they conclude that even simple prediction algorithms have a significant prediction power. We also consider their mechanism in the burstiness factors we define in Section 4. In [11], van de Meent et al. detect burstiness through the average traffic rate and the peak rate each second. They define a non-linear relation between these two variables to model the variation in the traffic rate that shows burstiness. Li et al. [12] detect burstiness in their model by defining thresholds. If the arrival rate exceeds the thresholds in a number of successive slots, then sustained burstiness occurs.

### 2.2. Burstiness detection based on TCP protocol

Burstiness has also been modeled for TCP traffic in other sudies such as like [13,14]. In [13], the authors detect bursts of TCP acknowledgment packet transmissions to increase the size of the congestion window. A burstiness model is defined in [14] that assigns a burstiness value to each TCP packet based on the RTT in order to control the actual sending rate.

### 2.3. Burstiness detection in databases

With regards to databases, Vlachos et al. [15] detect short-term and long-term bursts in online search queries by comparing the moving average (of 7 days and 30 days for short-term and long-term bursts, respectively) of user demands with its mean value.

### 2.4. Burstiness detection based on Internet traffic characterisation

Other papers deal with the characteristics of Internet traffic by focusing on the burstiness implicit to it. Sarvotham et al. [16] define an alpha/beta traffic model, considering the traffic bursts as the alfa-traffic and analyse why the bursts occur in network traffic. Lan et al. [17] define a burst as a train of packets with a lower inter-arrival time than a threshold and study the correlations between size, rate and burstiness.

As indicated above, we have included some ideas from previous papers [8–10] in our burstiness definition, which are detailed in Section 4. We have also used the non-linear relation defined in [11] to analyse some of the results obtained in Section 6.

## 3. Defining monitoring slots

The fact of introducing monitoring to a real system may introduce a relative error in the measurements taken. The most precise measuring of the system parameters could be carried out if monitoring is executed continuously and then all the changes in the system parameters are recorded in a database. The main problem involved is the extremely high overhead that is introduced in the system when many hundreds of requests per second arrive in the Web system and the monitoring process may modify the system's performance. Therefore, the monitoring itself may vary the values of the monitored parameters.

Moreover, obtaining average times of the values measured is a normal technique, even if all the system components are precisely and completely measured. This also leads to an implicit error in the accuracy of the final value of the observed metrics. We consider that this error cannot be avoided because without it, monitoring a system would get bogged down in the extraction and determination of minute details, which may make the overall analysis more difficult.

As previously stated, we propose monitoring the system by using adaptive time slot scheduling. To our knowledge there are no prior studies that set this scheduling to monitor the HTTP arrival rate at a Web system.

Obviously, an implicit error will be introduced. Let us analyse this fact with an example. Suppose we are monitoring the arrival rate of HTTP requests that arrive at the front-end of a Web system. It is evident that the arrival rate of the incoming requests is a random metric. If it is monitored by using fixed intervals, the resulting curve is different to the curve of the arrival rate monitored following adaptive time slot scheduling, which is due to the different values in the *x*-axis. Fig. 1 represents both curves and the monitoring intervals scheduled for each one. At the bottom of the plot, the observation times of the arrival rate monitored following a fixed time slot are illustrated. In this case the arrival rate is monitored every 20 s. At the top of the plot, the monitored time intervals are represented when following an adaptive time slot. It is clear that on average both arrival rate curves are equivalent but at each moment of the observation period one varies with respect to the other because of the different observation times. In Section 5, we explain how we set the duration of the adaptive monitoring time slots.

We are going to define the burstiness factors considered in this paper in order to study their behaviour, and then we detail how the adaptive time slot scheduling is defined.

## 4. Burstiness factors

We have considered six different approaches to define burstiness factors in order to compare their behaviour and detect their benefits or drawbacks under the same circumstances. Some of these approaches are based on previous articles we cited in Section 2 and the rest are modifications of them, as indicated below.

All the burstiness factor values are defined in [0,1] in order to limit the range of the factor because its value will be used later in the algorithm, for instance, to make some decisions to avoid a congestion situation. Hence, all the burstiness factors defined here can be easily adapted to be used in an admission control algorithm, despite being beyond the scope of this paper. The factor calculation is made for each time interval or slot defined by adaptive time slot scheduling, that is described in Section 5.

We start with the burstiness factor defined in [9] (BF1) and we propose some modifications to it to try to adapt it more accurately to the variations of incoming traffic (BF2, BF3, BF4). Some modifications are also included to introduce linear extrapolation in order to detect the bursty slots (BF5), as Baryshnikov et al. suggested in [10]. We have also considered a sixth burstiness factor (BF6) by including the proposal of Wang et al. [8]. In this case, the factor is computed for each incoming HTTP request to a Web server.

A description of each burstiness factor is given in the following subsections, and at the same time the reader can observe a representation of them in Fig. 2a–j. We have simulated all the burstiness factors proposed in this section in a discrete event simulator, OPNET Modeler [5]. The scenario consists of five Web servers that receive requests from 30 clients. The workload is fully described in Section 6. We show the behaviour of the factors we propose together with the arrival rate monitored in a Web server plotted on two scales (on the right the burstiness factor

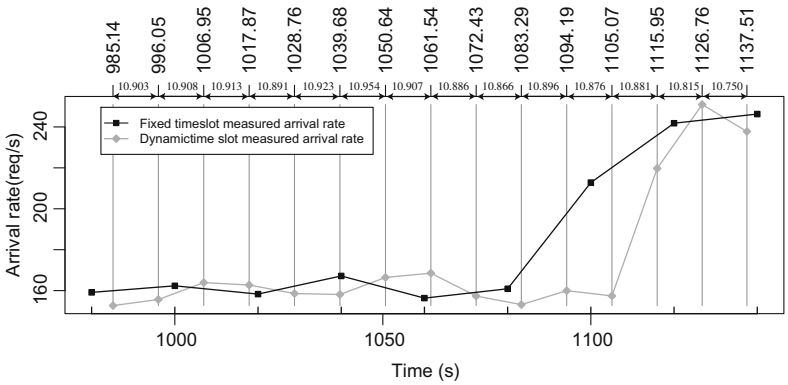


**Fig. 1.** Arrival rate monitored following different observation times.

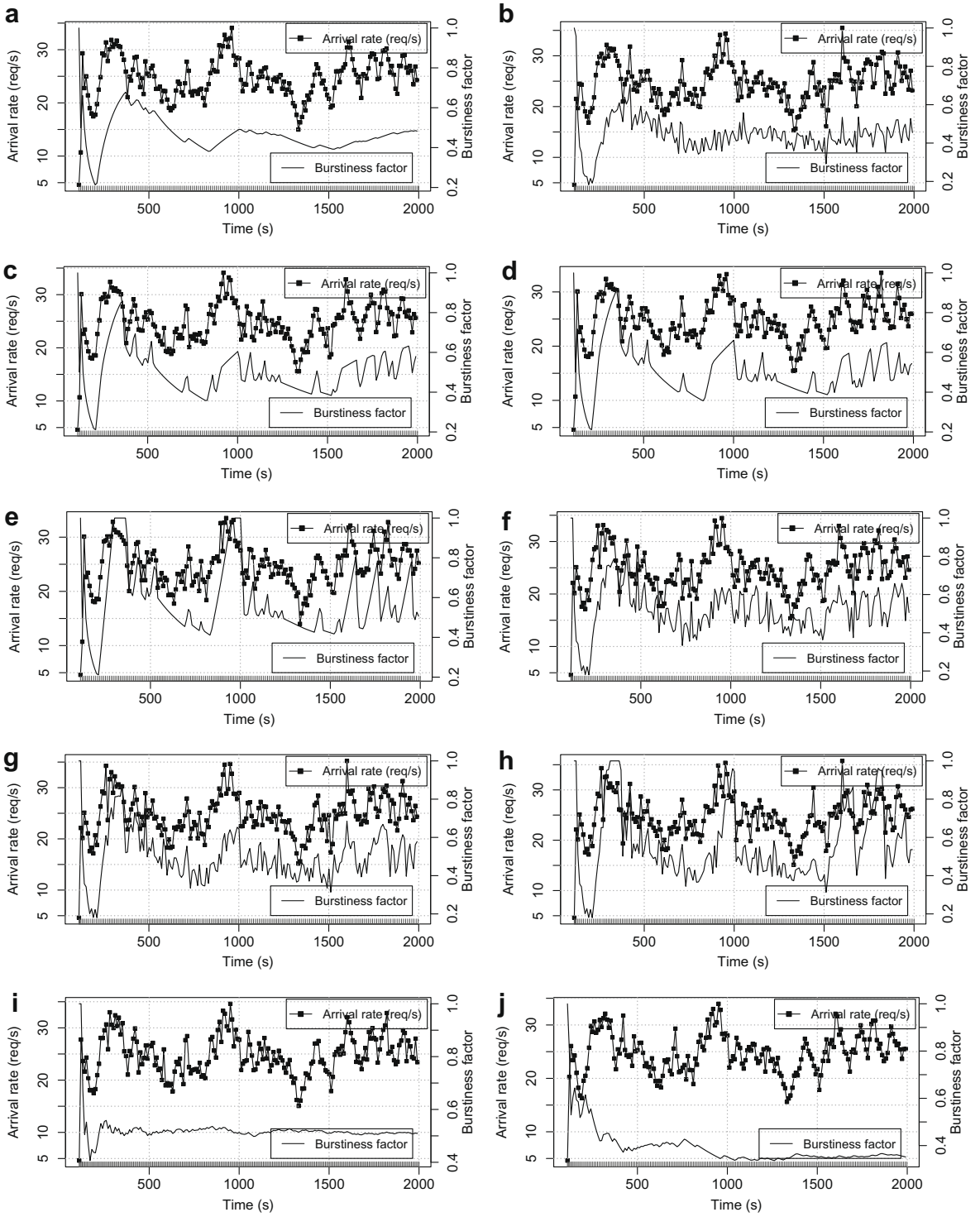


**Fig. 2.** Arrival rate and burstiness factors: (a) BF1; (b) BF2; (c) BF3 with $j = 3$; (d) BF3 with $j = 4$; (e) BF3 with $j = 10$; (f) BF4 with $j = 3$; (g) BF4 with $j = 4$; (h) BF4 with $j = 10$; (i) BF5; (j) BF6.

scale; on the left, the arrival rate scale). At the bottom of the figures the monitor time intervals of the arrival rate is given. All of the plots included in Fig. 2 have been obtained from the same simulation scenario that receive exactly the same workload, although some variations in the arrival rate can be observed. These variations are due to

the different lengths of the time intervals when we monitor the system, as explained in the previous section, which depends directly on the burstiness factor applied.

### 4.1. BF1: Menascé proposal

The first burstiness factor we use was proposed by Menascé and Almeida [9]. Its definition requires knowing the mean arrival rate of HTTP transactions for a Web server measured during some time intervals or slots $0, 1, \ldots, k-1$, denoted as $\mu_\lambda$. For a slot $k$ and a Web server, $\lambda(k)$ represents its corresponding arrival rate. If $\lambda(k) > \mu_\lambda$, then the slot is considered a *bursty* slot. Given $m$ slots, the burstiness factor is defined as the relation between the cumulative number of slots that satisfy $\lambda(k) > \mu_\lambda$, called $k^+$, and the current number of slots, $k$ [9]:

$$b_{orig}(k) = \frac{k^+}{k}. \tag{1}$$

This burstiness factor smooths the arrival rate curve. Fig. 2a illustrates that it follows the arrival rate but does not accurately represent its quick variations. We consider that the burstiness factor should alert the system as quickly as possible of an increase in the arrival rate, and this factor increases or decreases along with the increasing or decreasing arrival rate trend but very slowly and delayed.

We propose the direct inclusion of the arrival rate value in the burstiness factor in the next proposal, as a way to modify it quantitatively.

### 4.2. BF2: Arrival rate included

The second burstiness factor considered modifies the previous one by including the relative difference of the arrival rate of the two previous slots. Hence, the burstiness factor modification also depends on how much the arrival rate increases or decreases. Its expression is the following:

$$b_{arr\_rate}(k) = \frac{k^+}{k} \cdot \left(1 + \frac{\lambda(k) - \lambda(k-1)}{\lambda(k-1)}\right),$$
$$0 < b_{arr\_rate}(k) < 1. \tag{2}$$

Notice that $b_{arr\_rate}(k)$ can be greater than 1 in this definition. When this happens we set it to 1 to fulfill the $0 < b_{arr\_rate}(k) < 1$ restriction.

Fig. 2b shows that, in this case, the burstiness factor also varies with the variations of the arrival rate. Nevertheless, there are some peaks in the arrival rate that are not followed by the factor. In the next proposal we introduce a penalisation when detecting a consecutive number of *bursty* slots.

### 4.3. BF3: Penalisation included

We also want the burstiness factor to accurately represent the increasing traffic peaks incoming to a Web server. Hence, we consider that a maximum of $j$ consecutive *bursty* slots should cause a proportional increase in the burstiness factor. This is the reason for including a penalisation in the factor that depends on a record of previous *bursty* slots. This penalisation is limited with a record of $j$ slots, being $\alpha = 0.1 \cdot j$, for $j \in 1, \ldots, 10$. We have chosen to have a maximum record of 10 slots to penalise the burstiness factor because if the burstiness detected in the arrival rate is extreme, then the burstiness factor will be doubled every 10 slots. Hence, the maximum value of the burstiness factor can be easily reached:

$$b_{penalis(j)}(k) = \frac{k^+}{k} \cdot (1 + \alpha), \quad 0 < b_{penalis(j)}(k) < 1. \tag{3}$$

We have simulated the scenario with different values of $j$ to compare its behaviour. Fig. 2c–e represent the results obtained with this burstiness factor and a record of 3, 4 and 10 slots, respectively. We have omitted the plots corresponding to the rest of the values of $j$ in Fig. 2 because we consider that the behaviour of this factor is perfectly understood with these three values of $j$ and in this way we avoid the inclusion of more plots in this paper. It can be observed that as $j$ increases the burstiness factor penalisation also increases. We need to check if this penalisation (possibly excessive for high values of $j$) leads to an increase in the system performance or otherwise, decreases its performance because of an overreaction to the arrival rate.

### 4.4. BF4: Arrival rate and penalisation included

This proposal includes the relative arrival rate difference between the last two slots, which will permit the burstiness factor to follow all the arrival rate variations, and the penalisation:

$$b_{arr\_rate\_penalis(j)}(k) = \frac{k^+}{k} \cdot \left(1 + \frac{\lambda(k) - \lambda(k-1)}{\lambda(k-1)} + \alpha\right). \tag{4}$$

We also limit the value of $b_{arr\_rate\_penalis(j)}$:

$$0 < b_{arr\_rate\_penalis(j)} < 1.$$

The burstiness factor values in the simulation are shown in Fig. 2f–h, representing the results obtained with a maximum record of 3, 4 and 10 slots. We can observe that the resulting curves of BF4 are similar to the BF3 curves, but in this case the burstiness factor is also sensitive to changes in the arrival rate.

### 4.5. BF5: Linear extrapolation approach

In this case, we have used linear extrapolation of the arrival rate in order to detect bursty slots instead of the average of the arrival rate, as described by Baryshnikov et al. [10]. We compute the prediction of the arrival rate in the next slot for a Web server as the next expression, $t(k)$ being the final time of the slot $k$:

$$\hat{\lambda}(k) = \lambda(k-2) + \frac{t(k) - t(k-2)}{t(k-1) - t(k-2)} \cdot (\lambda(k-1) - \lambda(k-2)). \tag{5}$$

We consider a *bursty* slot when $\lambda(k) > \hat{\lambda}(k)$ and then we apply expression (1), considering $k^+$ as the number of *bursty* slots and $k$ as the current number of slots. This burstiness factor will be represented as $b_{extrap}(k)$.

Fig. 2i shows the results obtained with this burstiness factor and the resulting curve can be observed as being even smoother than the one obtained from the original Menascé proposal.

### 4.6. BF6: Wang approach

This last proposal, introduced by Wang et al. [8], has been considered in order to analyse its behaviour and compare it with the other proposals. The main difference between this proposal and all the previous ones is that this factor is computed for each incoming HTTP request that reaches a Web server.

Considering $t(1)$ and $t(m)$ as the times when packets 1 and $m$ arrive at the Web server, and $c(t(1),t(m)) = m-1$ as the number of packets between $t(m)$ and $t(1)$, the expression that Wang et al. used to represent the burstiness of each packet is the following [8]:

$$b(m) = t(1) + \frac{c(t(1),t(m))}{\rho_{min}} - t(m), \tag{6}$$

where $\rho_{min}$ represents the minimum throughput:

$$\rho_{min} = min\left[\frac{c(t(1),t(2))}{t(2)-t(1)}, \frac{c(t(1),t(3))}{t(3)-t(1)}, \ldots, \frac{c(t(1),t(m))}{t(m)-t(1)}\right].$$

In this case, burstiness is calculated in seconds and represents a measure of time that expresses the distance between the actual arrival time and the right edge of the $m$th packet arrival interval. As this definition of burstiness is expressed in seconds, we have modified it to be applicable to slots.

Hence, $\mu_b(k)$ being the mean of the burstiness of the packets that arrive in a slot $k$, and $\mu_b$ the mean of the burstiness of all the previous packets, we consider a *bursty* slot when $\mu_b(k) > \mu_b$ and then we apply expression (1), considering $k^+$ as the number of *bursty* slots and $k$ as the current number of slots. This burstiness factor will be represented as $b_{wang}(k)$.

In Fig. 2j, it can be observed that the BF6 curve does not accurately follow the arrival rate changes. The BF6 curve decreases in some points of Fig. 2j when the arrival rate curve increases. We will obtain more results with this burstiness factor in order to know its possible benefits with different workloads despite the fact that its calculation is made for each incoming HTTP request and then it needs a huge computational effort, which leads to a considerable overhead compared to the other proposals.

## 5. Adaptive time slot scheduling

Once we have defined the burstiness factors, let us describe how we use them to set up adaptive time slot scheduling.

We divide the total observation time $T$ of the experiment in several slots of variable duration. While the experiment is simulated, the duration of the slot changes based on the value obtained by the burstiness factor. Hence, the duration of the slot $k+1$ is dependent on the burstiness of the two previous slots, $b(k)$ and $b(k-1)$, as follows:

$$d(k+1) = \begin{cases} \frac{d(k)}{1+b(k)+b(k-1)}, & \text{if } b(k) \geqslant b(k-1), \\ \frac{d(k)}{1+b(k)-b(k-1)}, & \text{if } b(k) < b(k-1). \end{cases} \tag{7}$$

Therefore, the number of slots defined during the simulation time is also variable. We can calculate the total number of slots that divide the observation time $T$ during each slot. Considering the duration of the slot $k+1$, the frequency of slots is defined as

$$e(k+1) = \frac{T}{d(k+1)}.$$

The burstiness factor will never be 0 because we consider the first slot of the experiment as *bursty* to avoid a division by 0.

As the duration of the following slot is defined by the value of the burstiness factor on the current slot, when a burstiness increase is detected, the following testing time is brought nearer in order to check the incoming arrival rate early enough and then tune again the algorithm parameters. If a decrease in burstiness is perceived, the duration of the following slot is enlarged to reduce the overhead. By controlling the burstiness in the arrival rate, and then the duration of testing slots, a sudden reduction in the future performance of the Web servers may be forecasted.

An example of adaptive time slot scheduling is depicted in Fig. 3. In the upper part of the figure the arrival rate and the burstiness factor curve are drawn following adaptive time slot scheduling. As the arrival rate increases from time instant 910 s, the burstiness factor also increases. We have used BF1 to illustrate burstiness factor behaviour in this case. Below this figure, the slot duration is represented in another scale. It can be observed how the duration of the slots decreases when the arrival rate increases. Some slots have been zoomed in to detail the decrease of their durations.

## 6. Simulation scenario and results

We have tested the six burstiness factors described in Section 4 in the discrete event simulator OPNET Modeler [5]. The architecture is made up of a set of clients that request Web contents from the Web system as depicted in Fig. 4. The load balancing algorithm employed in this scenario is fully described in [7] and is beyond the scope of this document. The architecture is modeled as a one-way architecture that provides an alternative way for the responses to reach the clients, instead of going through the load balancer, in order to prevent the load balancer from becoming the system bottleneck.

We have considered 30, 40 and 50 clients in order to compare the results obtained by the six burstiness factors. Each of the tests has been simulated with four seeds for 5, 10, 15 and 20 Web servers in the system.

The workload generated by the set of Web clients contains dynamic and static HTTP requests to the Web cluster and has been modeled according to recent results in Internet traffic literature. We have considered four types of applications that can be executed concurrently by the Web clients. Each of these applications asks for Web content with a user think time that follows a Pareto distribution $(k = 0.3, \alpha = 1.4)$ [18,19]. As our intention is to stress the system with intense workload by using a low

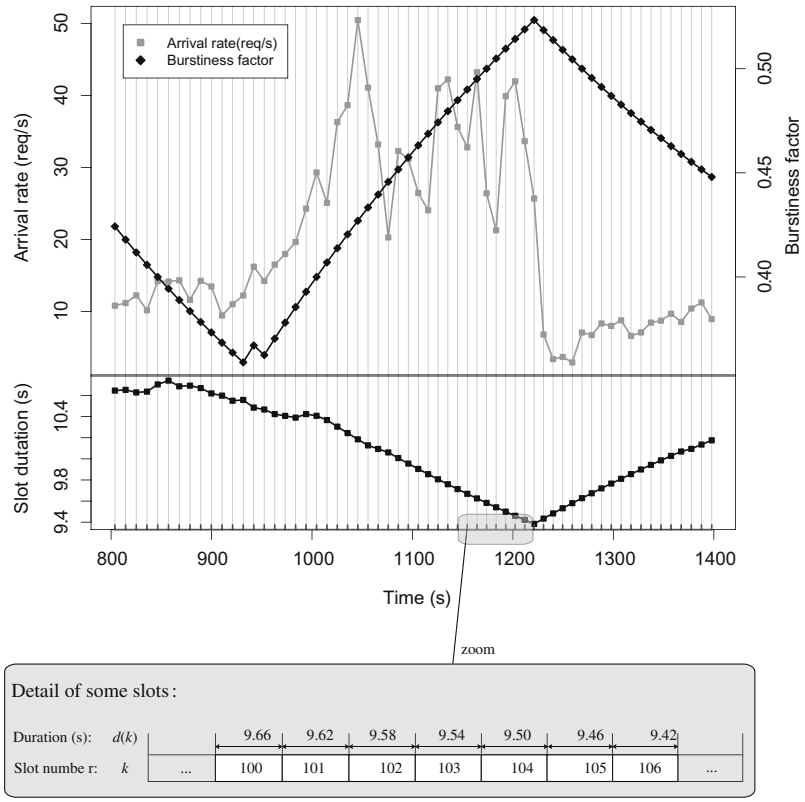


**Fig. 3.** Arrival rate monitored following adaptive time slot scheduling and detail of some of the slots using the BF1.

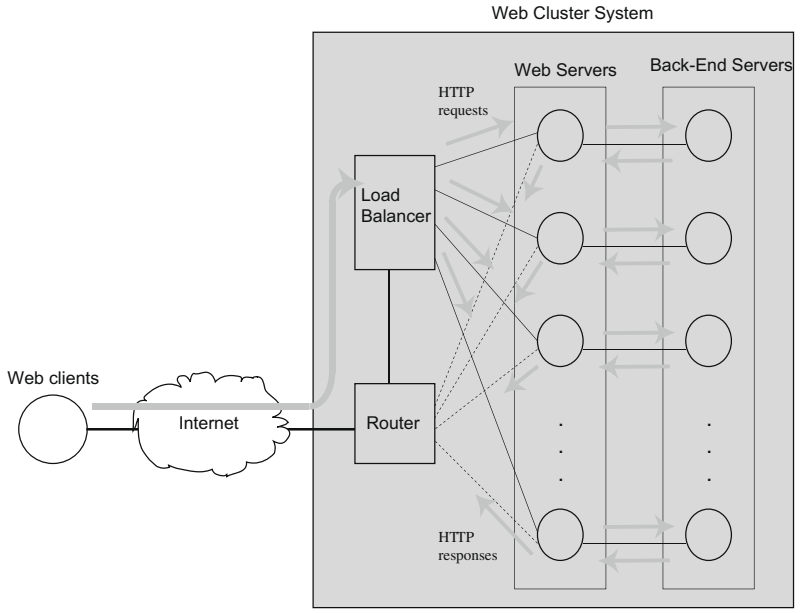


**Fig. 4.** The Web architecture is made up of 20 mirrored Web servers and their corresponding database servers. All the Web servers are connected to a load balancer that distributes the load. The model architecture is one-way, which means that the incoming HTTP requests go through the load balancer but their HTTP responses use a different way to prevent the load balancer from becoming the system bottleneck.

number of clients in order to simplify the scenario design, the equivalent of 30, 40 and 50 clients requesting for Web content with four concurrent application means that an average of 125, 160 and 200 requests per second reach

the Web system. The session duration and the session inter-repetition time are modeled according to an exponential distribution, as it is documented in [20,21], respectively.

The size of the Web pages the clients ask for has been obtained from the logs of the 24th of June of the 1998 World Cup Web Site [22]. Arlitt et al. [1] estimate, after analysing these logs, that the body of the unique file size distribution follows a lognormal distribution and also that it is heavy tailed. A summary of the workload specification is shown in Table 1.

Two experiments have been carried out in the scenario shown in Fig. 4. The first experiment stresses the system during a total simulation time of 2000 s with the workload specified in Table 1. The second experiment considers an increase in the arrival rate at 1000 s of the simulation time and for 200 s by changing the user think time from a Pareto ($k = 0.3, \alpha = 1.4$) distribution to a Pareto ($k = 0.2, \alpha = 1.4$). This means an important increase in the traffic that arrives at the system. The purpose of this modification is to analyse how the burstiness factors react to a sudden increase of the arrival rate.

### 6.1. First experiment: no changes in the workload

As we want to know the relation between the arrival rate and the burstiness factor detected in each server, we have chosen to compute their correlation by using the standard Pearson method [23]. In the central set of columns, Table 2 shows the maximum of the 95th percentile of the correlation values between the arrival rate and the burstiness factors detected in the servers for 30, 40 and 50 clients in this first experiment. We have considered the differences of the values of each statistic in two consecutive slots because this is the way we have formulated most of the expressions above. The correlation values between the arrival rate and the slot frequency are very similar to those presented in this table due to the definition of the slot frequency, which is directly dependent on the burstiness factor, hence we have omitted it.

**Table 1**
Workload specification.

| Number of Web servers | 5, 10, 15, 20 |
|---|---|
| Number of clients | 30, 40, 50 |
| File size | Lognormal body and heavy tailed |
| User think time (s) | Pareto ($k = 0.3, \alpha = 1.4$) |
| User session duration (s) | Exponential ($\mu = 600$) |
| Session inter-repetition time (s) | Exponential ($\mu = 30$) |

**Table 2**
Maximum correlation between the 95th percentile of the differences of the burstiness factor and the arrival rate in two consecutive slots for (i) 30 clients, (ii) 40 clients and (iii) 50 clients.

| | First experiment | | | Second experiment | | |
|---|---|---|---|---|---|---|
| | (i) | (ii) | (iii) | (i) | (ii) | (iii) |
| BF1 | 0.3433 | 0.3840 | 0.4078 | 0.2406 | 0.2417 | 0.2033 |
| BF2 | 0.8205 | 0.8195 | 0.8141 | 0.7698 | 0.7263 | 0.7361 |
| BF3 | | | | | | |
| $j = 3$ | 0.5386 | 0.5677 | 0.5395 | 0.6940 | 0.6928 | 0.6764 |
| $j = 4$ | 0.5267 | 0.5187 | 0.5467 | 0.7375 | 0.6848 | 0.6579 |
| $j = 10$ | 0.5011 | 0.4862 | 0.5273 | 0.5996 | 0.6263 | 0.6550 |
| BF4 | | | | | | |
| $j = 3$ | 0.8711 | 0.8508 | 0.8495 | 0.8123 | 0.8287 | 0.8330 |
| $j = 4$ | 0.8551 | 0.8393 | 0.8469 | 0.8513 | 0.8422 | 0.8390 |
| $j = 10$ | 0.8225 | 0.7843 | 0.7974 | 0.8160 | 0.8395 | 0.8344 |
| BF5 | 0.6313 | 0.6168 | 0.5735 | 0.4497 | 0.4663 | 0.4326 |
| BF6 | 0.1179 | 0.1328 | 0.0798 | 0.1585 | 0.1444 | 0.1260 |

We find that both statistics are strongly correlated when the used burstiness factor includes the arrival rate in its formula, which is the case of BF2 and BF4. Indeed, BF4 is the most correlated, but its correlation decreases as $j$ increases (in all cases: (i)–(iii)). The same occurs with BF3. This means that these burstiness factors are probably excessively penalised when $j$ is increased for the proposed workload.

Comparing columns (i)–(iii) in Table 2 for the first experiment we can observe that, despite the fact that its correlation values are less than 0.5, BF1 improves its maximum correlation when more traffic reaches the server. The rest of the burstiness factors do not show this improvement in the first experiment.

In order to obtain a deeper understanding of the benefits of each burstiness factor, Fig. 5 shows the relation between the differences of the 95th percentile of the arrival rate values and the slot frequency in two consecutive slots for 30 clients. A smooth curve computed by Loess [24] has been added to the plots in Fig. 5 to highlight the trend of this relation for 5, 10, 15 and 20 servers. As the workload generated by the 30 clients is the same in all the cases, when we have five active servers in the Web system, more requests per second arrive at each server than if we have 20 active servers. Hence, the Loess smooth curve differentiates the arrival rate and slot frequency changes for each case.

It can be observed that BF1, BF5 and BF6 (see Fig. 5a, i and j, respectively), scarcely modify the values of slot frequency when changing the arrival rate difference for 5, 10, 15 and 20 servers. The case of BF3 (see Fig. 5c, d and f) shows a bent Loess curve because when a non-bursty slot is detected after a sequence of bursty slots, the burstiness factor decreases suddenly returning to its original value. This can also be observed in Fig. 2. In the case of BF4, there is also a decrease in the burstiness factor when a non bursty slot is detected, but it is smoother because it depends on the arrival rate detected in the server. The clearer linear relation is obtained with BF2 and BF4 for 5, 10, 15 and 20 servers, which supports the results obtained by Table 2.

### 6.2. Second experiment: increasing the workload

In order to contrast the results obtained in the first experiment, we have increased the workload significantly to check whether the correlation values are maintained by the burstiness factors studied in this second experiment. A general reduction can be observed of the correlation values in Table 2, in the right set of columns, due to the peak in the arrival rate we introduce when we modify the user think time at 1000 s. This is not the case for BF3, which increases its $j$ values and seems to adapt better to this sudden change in the arrival rate. Nevertheless, BF2 and BF4 are still the most correlated.

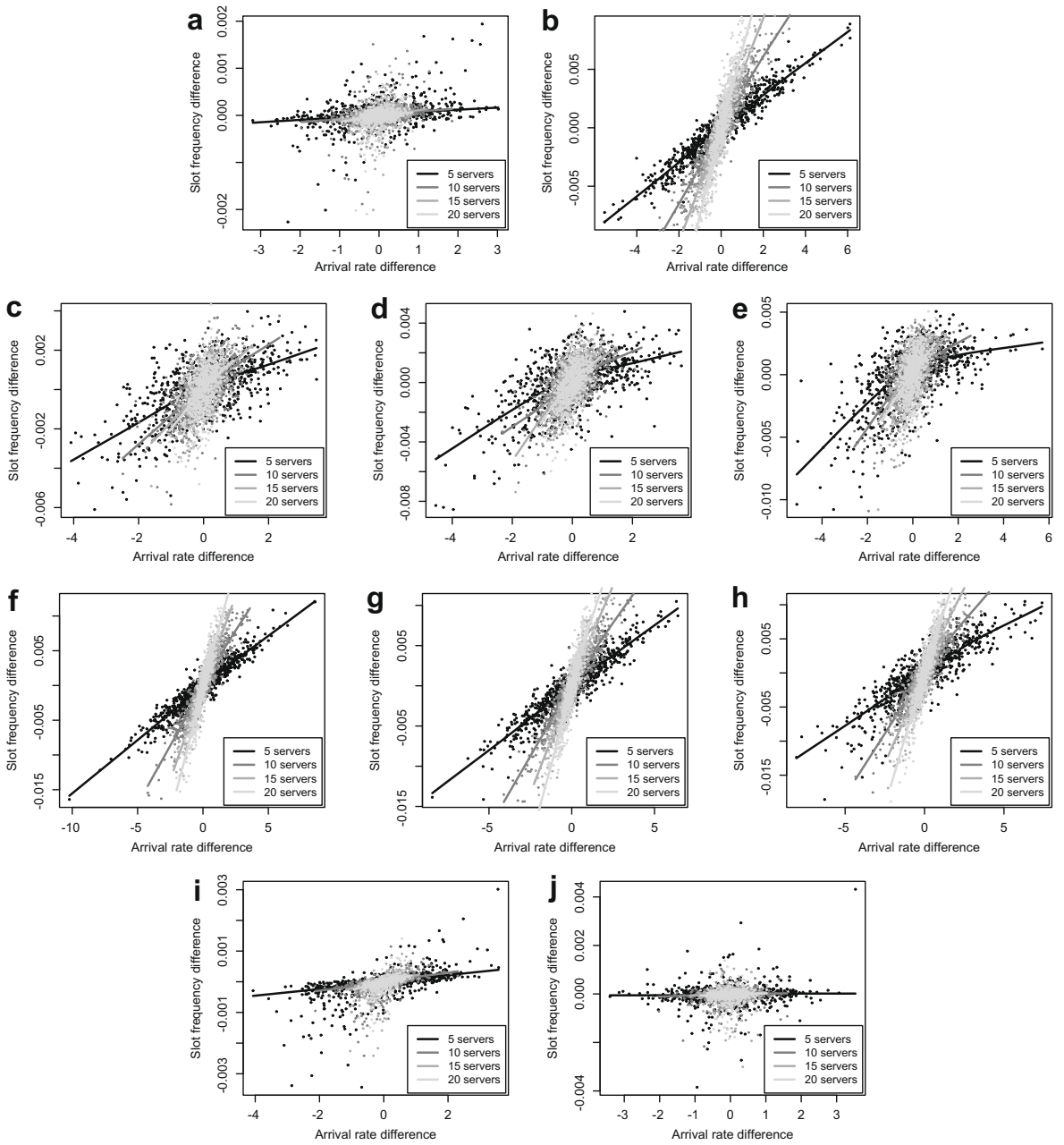


**Fig. 5.** First experiment: no changes in the workload for 30 clients. Linear relation between the arrival rate and the frequency of slots considering different burstiness factors: (a) BF1; (b) BF2; (c) BF3 with $j = 3$; (d) BF3 with $j = 4$; (e) BF3 with $j = 10$; (f) BF4 with $j = 3$; (g) BF4 with $j = 4$; (h) BF4 with $j = 10$; (i) BF5; (j) BF6.

For a better analysis of the results, we have related the average and peak traffic rates (see Fig. 6a) as van de Meent et al. describe in [11]. The objective is to compare this traffic rate relation with the relation of average and peak burstiness factor values for each of the six burstiness factors considered, highlighting the results obtained for 5, 10, 15 and 20 servers (see Fig. 6b–k). The more similar the traffic rate relation figure and the burstiness factor relation figure are, the better the burstiness factor represents the variations of the arrival rate and consequently adapts its value.

The comparison shows an excessive penalisation on BF3 and BF4 for all the values of $j$. The reason is that when a large number of consecutive slots are bursty, then these factors increase their own values more and more. Hence, they easily reach their maximum values (that is 1) and remain near it most of the simulation time. The greater the $j$ of BF3 and BF4 is, the more times they reach 1. This makes it almost impossible for them to detect a sudden, even greater peak in the arrival rate reaching the system because most of the times these factors have reached their maximum values.

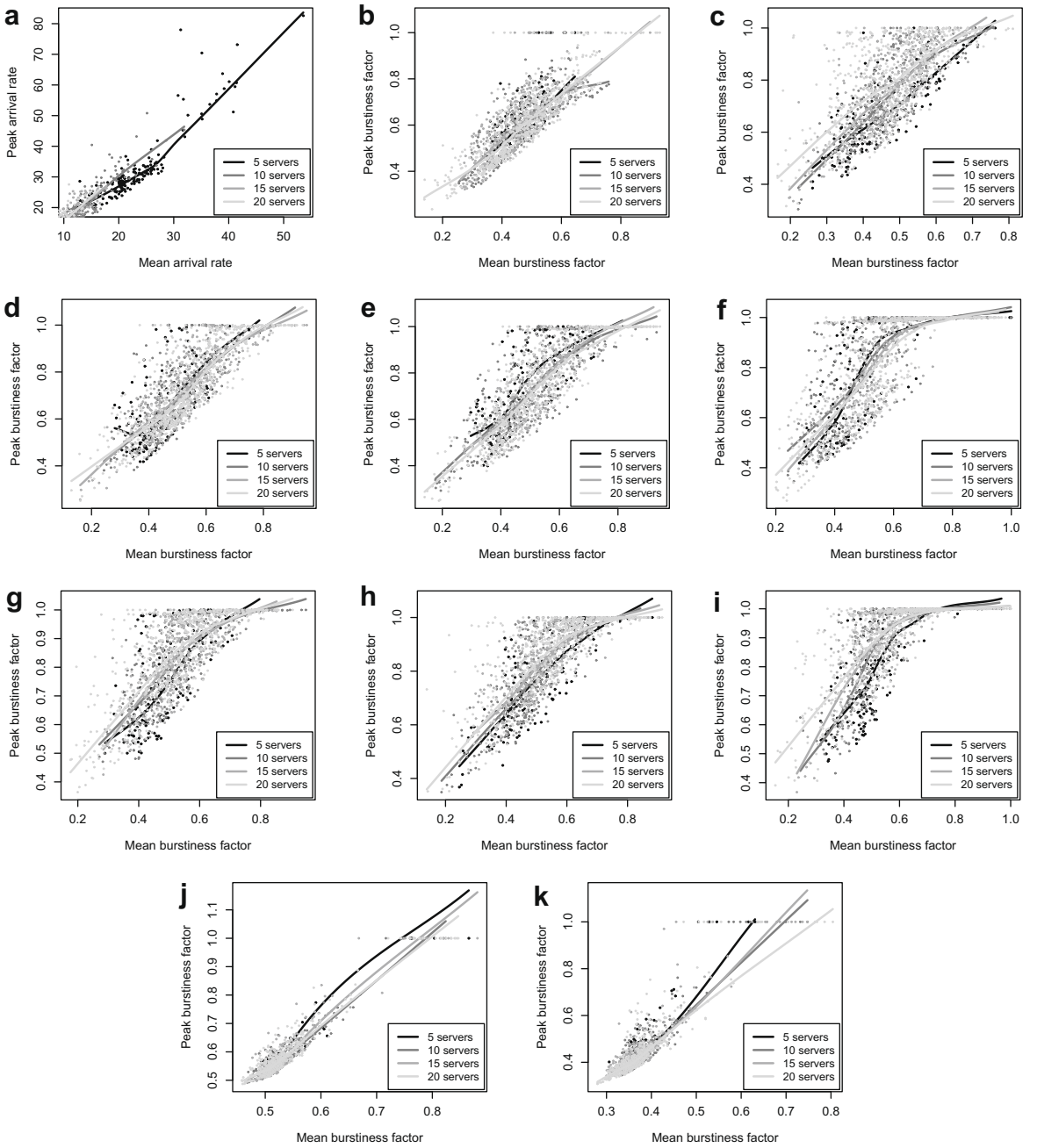


**Fig. 6.** Second experiment: increasing the workload for 30 clients. (a) Mean and peak traffic rates for each Web server. Mean and peak burstiness factors for each Web server: (b) BF1; (c) BF2; (d) BF3 with $j = 3$; (e) BF3 with $j = 4$; (f) BF3 with $j = 10$; (g) BF4 with $j = 3$; (h) BF4 with $j = 4$; (i) BF4 with $j = 10$; (j) BF5; (k) BF6.

BF1, BF4 and BF5 (Fig. 6b, j and k, respectively) reach their maximum values of 1 fewer times during the simulation time of this second experiment. We can label them as the most conservative because they react to the arrival rate variations by slightly changing their values, which is a drawback if there is a sudden increase in the arrival rate.

BF2 shows almost the same behaviour for 5, 10, 15 and 20 servers (see Fig. 6c) compared to the variations of the arrival rate plot for a different number of servers (Fig. 6a), due to the fact that BF2 adaptively changes with the variations in the workload in each case. However, the comparison among the different number of servers in not very important because their results correspond to different simulations.

## 7. Conclusions

The aim of this paper is to study several burstiness factors that detect the variations in the arrival rate at a Web

system. This leads to a monitoring scheduling that is defined by adaptive time slot scheduling to minimise the execution overhead of the monitoring itself and the actions of the algorithm that uses the monitored information. If the burstiness factor detects an increase in the arrival rate trend to the system, the adaptive time slot scheduling proposed permits checking times are closer in the terms of time, or viceversa if an arrival rate decrease is detected. The set of the slot duration is calculated based on the burstiness factor, enabling the adaptive monitoring of the bursty arrivals at the system.

Six different burstiness factors have been considered in this work. The main advantage of defining a burstiness factor is that is can be located in any part of the Web system that receives an arrival rate. We have chosen to detect the burstiness in the Web servers of the Web system. Some of the burstiness factors defined (BF1, BF5 and BF6) are conservative in the sense that they do not change their values significantly with the arrival rate variations; while others (BF3 and BF4) include some penalisation when detecting successive bursty slots that force them to change their value considerably. We have found that a burstiness factor that quickly follows changes in the arrival rate trend is good enough as long as its maximum value is not easily reached. Conservative burstiness factors are not very useful if there is a sudden increase in the arrival rate.

An accurate tracking of the arrival rate trend by the burstiness factor is mandatory. It is also important to choose a burstiness factor that permits detecting an increase in the arrival rate independently of the actual workload in the system. These are the main reasons for us to conclude that BF2 seems the best candidate for an adaptive Web system.

An open problem in the design of a burstiness factor that includes a penalisation is the decision of the penalty amount to be included in it. Especially in the case of detecting several consecutive bursty slots. The factor has to be suitable for all possible arrival rates a Web system may expect and has to follow arrival rate variations accurately in the range of values defined for it.

The impact of the different burstiness factors in the performance of the load balancing algorithm is also an open question to be analysed, which could be included in a future analysis of burstiness factors in a globally distributed Web server architecture versus a locally distributed Web server architecture.

Also, as a further study we would like to consider that traffic differentiation in the burstiness factor may improve the performance of the Web system as not all types of traffic demand the same service in the Web servers.

## Acknowledgements

This work has been partially funded by the spanish Ministry of Education and science under Grant TIN2006-02265. The authors would like to thank Joris Slegers for his helpful comments.

## References

[1] M. Arlitt, T. Jin, A workload characterization of the 1998 World Cup Web site, IEEE Network (2000) 20–37.
[2] M. Meiss, F. Menczer, A. Vespignani, On the lack of typical behavior in the global Web traffic network, in: Proceedings of the World Wide Web, 2005, pp. 510–518.
[3] G. Banga, P. Druschel, Measuring the capacity of a web server under realistic loads, in: Proceedings of the World Wide Web, 1999, pp. 69–83.
[4] Q. Zhang, A. Riska, E. Riedel, Workload propagation overload in bursty servers, in: Proceedings of the QEST, 2005, pp. 1–10.
[5] Opnet Technologies Inc. <http://www.opnet.com/>.
[6] K. Gilly, S. Alcaraz, C. Juiz, R. Puigjaner, Burstiness detection in a web adaptive cluster system, in: Proceedings of the Industry Track to First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTOOLS), 2008.
[7] K. Gilly, S. Alcaraz, C. Juiz, R. Puigjaner, Service differentiation and QoS in a scalable content-aware load balancing algorithm, in: Proceedings of the 40th Annual Simulation Symposium, 2007, pp. 185–193.
[8] Z. Wang, J. Crowcroft, Analysis of burstiness and jitter in real-time communications, in: Proceedings of the SIGCOMM, 1993, pp. 13–19.
[9] D.A. Menascé, V.A. Almeida, Capacity Planning for Web Performance, Metrics, Models and Methods, Prentice Hall, 1998.
[10] Y. Baryshnikov, E. Coffman, D. Rubenstein, B. Yimwadsana, Traffic prediction on the internet, Technical Report EE200514-1, Computer Networking Research Center Columbia University, 2002.
[11] R. van de Meent, A. Pras, M. Mandjes, J. van den Berg, F. Roijers, L. Nieuwenhuis, P. Venemans, Burstiness predictions based on rough network traffic measurements, in: Proceedings of the 19th World Telecommunications Congress (WTC/ISS), 2004.
[12] H. Li, C. Huang, M. Devetsikiotis, G. Damm, Effective bandwidths under dynamic Weighted Round Robin scheduling, in: Proceedings of the IEEE GLOBECOM, 2004, pp. 665–669.
[13] M. Gerla, M.Y. Sanadidi, R. Wang, A. Zanella, TCP westwood: congestion window control using bandwidth estimation, in: Proceedings of the IEEE GLOBECOM, 2001, pp. 1698–1702.
[14] D.X. Wei, S.H. Low, A burstiness control for TCP, in: Proceedings of the Workshop on Protocols for Fast Long-Distance Networks, 2005.
[15] M. Vlachos, C. Meek, Z. Vagena, D. Gunopulos, Identifying similarities, periodicities and bursts for online search queries, in: Proceedings of the SIGMOD, 2004, pp. 131–142.
[16] S. Sarvotham, R. Riedi, R. Baraniuk, Connection-level analysis and modeling of network traffic, in: Proceedings of the SIGCOMM, 2001.
[17] K.-C. Lan, J. Heidemann, A measurement study of correlations of Internet flow characteristics, Computer Networks 50 (1) (2006) 46–62.
[18] P. Barford, M. Crovella, A performance evaluation of Hyper Text Transfer Protocols, Measurement and Modeling of Computer Systems, 1999, pp. 188–197.
[19] E. Casalicchio, V. Cardellini, M. Colajanni, Content-aware dispatching algorithms for cluster-based web servers, Cluster Computing 5 (2002) 65–74.
[20] L. Cherkasova, P. Phaal, Session-based admission control: a mechanism for peak load management of commercial web sites, IEEE Transactions on Computers 51 (2002) 669–685.
[21] T. Bonald, J. Roberts, Congestion at flow level and the impact of user behaviour, Computer Networks 42 (2003) 521–536.
[22] The internet traffic archive. <http://ita.ee.lbl.gov/>.
[23] J.L. Rodgers, W.A. Nicewander, Thirteen ways to look at the correlation coefficient, The American Statistician 42 (1988) 59–66.
[24] W.S. Cleveland, Robust locally weighted regression and smoothing scatterplots, Journal of the American Statistical Association 74 (1979) 829–836.

**Katja Gilly** graduated from the University of Alicante in 1995 and received her M.Sc. in Computer Science in 1998. She worked from 1998 till 2001 for IBM in the OS/390 networking group. In 2001, she joined the Department of Physics and Computer Architecture in the Miguel Hernandez University. She is actually working in her Ph.D. in the Department of Mathematics and Computer Sciences (University of Balearic Islands) about Web performance and has published several papers on this subject. She has participated in several national and international research projects and is IEEE and ACM member since 2006.

**Salvador Alcaraz** graduated from University of Alicante in 1995 and received his M.Sc. in Computer and Automation from Miguel Hernandez University in 2000. In this year, he joined the Department of Physics and Computer Architecture in the same university. He is actually working in his Ph.D. in the University of Balearic Islands about QoS in Web traffic. Recently, he has published several papers, attended international conferences and participated in international research projects.

**Carlos Juiz** received the B.Sc., M.Sc. and Ph.D. degrees in Computer Science in 1988, 1992 and 2001, respectively, from the University of Balearic Islands (UIB), Spain. He has a postgraduate degree on Office automation from the Polytechnic University of Madrid, Spain (1993). Before joining as Assistant Professor in the Department of Mathematics and Computer Sciences (UIB), he had several positions related with the software industry. From 1990, he was Systems Analyst at Xerox, leaving this position as Senior Analyst in 1999. He was visiting researcher at Department for Computer Science and Business Informatics, University of Vienna, during 6 months in 2003. His research interest mainly focuses on systems performance modelling, Web performance engineering and ambient intelligence middleware. He has been involved in several Spanish and international research projects. He has participated in numerous international conferences, workshops and congresses as program committee member. He is co-author of more than 80 international publications (20 journals, 18 reviews, 44 proceedings and 3 monographic book chapters) and one book. He was editor of special issues in international journals and permanent reviewer of several indexed publications. He was Sub-director of the Polytechnic School of the University of Balearic Islands (UIB, 2004–2005) and Director of the Office of Planning and Prospective (UIB, 2005–2007). He is currently the Chancellor's Delegate for new technologies at UIB.

**Ramon Puigjaner** received the degree of Industrial Engineer from the Universitat Politècnica de Catalunya (Barcelona, Spain) in 1964, his Master degree in Aeronautical Sciences from the Ecole Nationale Supérieure de l'Aéronautique de Paris (France), his Ph.D. from the Universitat Politècnica de Catalunya (Barcelona, Spain) in 1972 and his degree of License in Informatics from the Universidad Politécnica de Madrid (Spain) in 1972.

From 1966 to 1987, he shared his time between the Polytechnic University of Catalonia, where he taught and researched on Automatic Control, Computer Architecture and Computer Performance Evaluation, and several positions in the industry, mainly from 1970 to 1987 at UNIVAC (after SPERRY and finally UNISYS) where he was in charge of computer performance measuring and modelling for tuning and sizing in Spain. In 1987, he joined the Department of Computer Science of the University of Balearic Islands (Palma de Mallorca, Spain) full time, where he is currently Professor of Computer Architecture and Technology.

He is author of a book on computer performance evaluation and of more than 150 reviewed papers in international journals and conferences.

His current research interests are the performance evaluation of computer systems and computer networks and the diffusion of these techniques in the industrial milieu mainly in the field of real-time and distributed systems and ambient intelligent systems.

He has been involved in several ESPRIT projects and is still involved in several projects funded by the Spanish Comisión Interministerial de Ciencia y Tecnología. He has acted as project reviewer and evaluator for the Commission of the European Union.