Server Load Estimation by Burr Distribution Mixture Analysis of TCP SYN Response Time

Luis de Pedro[a,b,*], Adrian Mihai Rosu[b], Jorge E. López de Vergara[a,b]

[a] *Department of Electronic and Communication Technologies, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Spain*
[b] *Naudit High Performance Computing and Networking, S.L., Spain*

Abstract

Server load estimation is key in balancing traffic between servers when optimizing data center resources. Intrusive methods are sometimes difficult or impossible to implement. Therefore, non-intrusive estimation methods are the best alternative in these cases. The objective of this paper is to present a server load estimation method based on external network traffic measurements obtained in a vantage point close to the server. Statistical distributions of TCP SYN response time, that is, the time from SYN to SYN+ACK segments at the server side, are used to fit Burr Type XII heavy tail distribution mixtures. The fitting algorithm, based on maximum likelihood estimation, is developed in detail in this paper. Experimental data shows that the median of the fitted distribution correlates within the 95% confidence interval of the server load figures and, thus, it can be used as a non-intrusive and accurate method to measure it. This new method can be applied to almost any existing load balancing algorithm, as it does not make any assumption about the server, which is considered a black box.

Keywords: non-intrusive measurement, server load, Burr Type XII distribution, MLE, mixture analysis, traffic analysis.

1. Introduction

Server load can be defined as the percentage of use of available computational resources in that system. Usually, it is directly measured with software tools inside the server (e.g., with `mpstat` or with an SNMP agent). However, there are circumstances in which the load cannot be measured this way. For instance, when the access to the operating system is not available to the user (as in a Platform-as-a-Service deployment). In fact, server load estimation is a common issue in many Data Centers [1], as it is convenient to balance jobs between different systems. Occasionally, it is mandatory to fix bottlenecks in performance. Moreover, server load can also be useful to address other problems that have appeared in distributed systems recently, such as cloud computing infrastructures. In these environments, physical server load is not always available from the Cloud services provider, providing virtual server load instead [2]. Commercially available servers usually provide tools to estimate server load, but they need access to administration capabilities and may worsen an excessive load problem. Moreover, a local tool may provide wrong results if the server is a virtual machine hosted in a cloud infrastructure [3].

Server load estimation is a topic frequently addressed in the literature in the context of load balancing. There are comprehensive surveys [4, 5, 6] and comparisons [7]. Many of them are based on non-intrusive estimations of server availability [8] or, more often, in server load measurements [7, 9, 10]. Therefore, server load data availability has crucial importance to operate clusters efficiently. On the other hand, server load is the input to a number of proposed scheduling algorithms [11, 12] and key to distinguish if a problem is caused by the network or by the server. This helps to reduce the time to identify its root causes, or, at least, the so-called mean time to innocence (MTTI) [13]. Even mobile services efficiency algorithms rely on server load data to avoid systems saturation [14]. Besides that, IT energy consumption is one of the few ones that can be moved from one Data Center to another in some other region. This helps to optimize energy consumption for reducing the carbon footprint [15, 16]. In that sense, this work may be a modest contribution to climate change fighting.

As shown, server load estimation without actual system access is desirable in several scenarios, although we should look out of the server. We can guess that response time to network protocols may be related to system load. If so, when network traffic is available for analysis, the goal of server load estimation can be achieved, as we will show in this paper.

Network traffic has been widely monitored and analyzed to study the network behavior, with many research papers [17, 18], as well as several patents about this topic [19, 20]. Use of statistics has been proposed to estimate parameters like bandwidth availability [21]. Moreover, it can also be useful to study end systems behavior. For instance, several zero-window announcements in a TCP session show that an application is receiving data at a higher rate than the one it can process. Such a type of information can be very useful to identify the root cause

[*] Corresponding author

Email addresses: `luis.depedro@uam.es` (Luis de Pedro), `adrian.rosu@naudit.es` (Adrian Mihai Rosu), `jorge.lopez_vergara@uam.es` (Jorge E. López de Vergara)

Received: 6th June 2022. Revised: 28th November 2022. Revised: 27th February 2023. Accepted 28th June 2023.

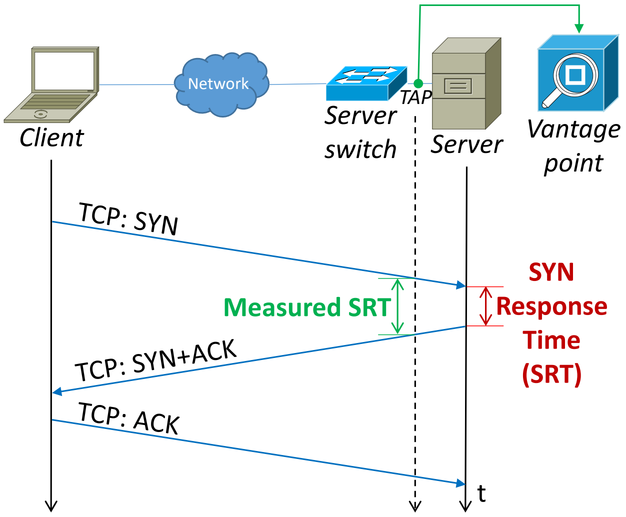

Fig. 1. SYN Response Time in the TCP handshake and its measurement at a vantage point close to the server.

of problems [22] in distributed systems, which are usually difficult to fix.

Thus, to estimate server load without measuring it internally, we propose a passive approach using traffic analysis. Our starting point is traffic capture using a non-intrusive probe (a network TAP, for instance) close to a server. By "close" we mean there is no active communications equipment (switches or routers) between the probe and the target system that could contaminate the measurement, due to interfering traffic or queuing delays.

Our previous work [23] has shown a strong correlation between the response time to a SYN segment (SRT, SYN Response Time) and the load on a given server. There, we presented how an SRT distribution is modified when server load increases. Moreover, it has also been proved that heavy-tailed α-stable distributions can be used to fit experimental data with promising accuracy.

In this paper, those previous results are further refined. We confirm the strong correlation between the measured SRT and the server load. Furthermore, we analyze the actual SRT distribution and derive a distribution mixture fit whose statistics have proven useful to estimate server load. One of the problems found in our previous work is the complexity of the SRT statistic distributions. We noticed that a single mode distribution is not accurate enough. Instead, a mixture of distributions is much more precise. However, α-stable distribution mixture fitting is quite computing intensive. Several attempts to address the calculation issues in real environments are available [24, 25] but the main problem stands: α-stable distribution probability density function (PDF) does not have a closed expression [26]. This point makes it very cumbersome to estimate the parameters. On the other hand, Burr Type XII distributions have an explicit expression and have been used to model different environments [27, 28, 29]. We have verified that, using Burr distributions mixtures, the SRT data can be modeled. Here, we present a method to estimate the parameters. In this case, the calculations are much easier and the accuracy is better than in our previous α-stable approach [23].

The main contribution of this paper is a novel method to estimate the server load using a network-traffic-based, non-intrusive approach by analyzing the SRT distribution mixture characteristics. Moreover, other relevant contributions also follow:

1. It is demonstrated that the SRT distribution can be accurately modeled by using a Burr Type XII distribution mixture to fit the multimodal heavy-tailed nature of the SRT. In contrast, other typical distributions do not fit well or are difficult to calculate.

2. A generalized method is presented to fit experimental data with a Burr Type XII mixture. It is independent of the number of distributions that compose the mixture. Previous works could only calculate a mixture of two or three Burr Type XII distributions [30, 31].

3. The relation between distribution statistics and the server load is analyzed. Based on this relation, we obtain a curve by regression that predicts the server load from the median of the SRT distribution.

4. Finally, this new method can be applied to almost any existing load balancing or scheduling algorithm, as it does not take any assumption about the server, which is considered a black box.

The rest of the paper is organized as follows. First, related work is presented. Next, the SRT concept is reviewed to focus on the network analysis framework. Then, the methodology based on Burr Type XII mixture PDF fit to measurements is presented. Next, the mathematical model is developed, describing the generalized method to fit a Burr Type XII mixture. Next, the proposed algorithm is used to predict server load with actual data. Later, results and limitations are discussed. Finally, conclusions are provided.

2. Related Work

From the best of our knowledge, apart from our previous work at [23], there is no other work that uses the SRT to estimate the server load. In this section, we mention other related works that have estimated server load. As stated before, it can be difficult to measure server load without interfering significantly with the applications running on it. Therefore, several non-intrusive approaches have been proposed.

Indirect estimation using Autoregressive Integrated Moving Average Model (ARIMA) is proposed in on-demand resources provision algorithms [9] to optimize data migration in a cluster of servers. The load estimation figure is then used to schedule resource provision and data migration between clouds. This method avoids late reaction when resources are being overwhelmed. Early data migration is crucial to reduce services expenditure and load balancing in a cloud. That method is based

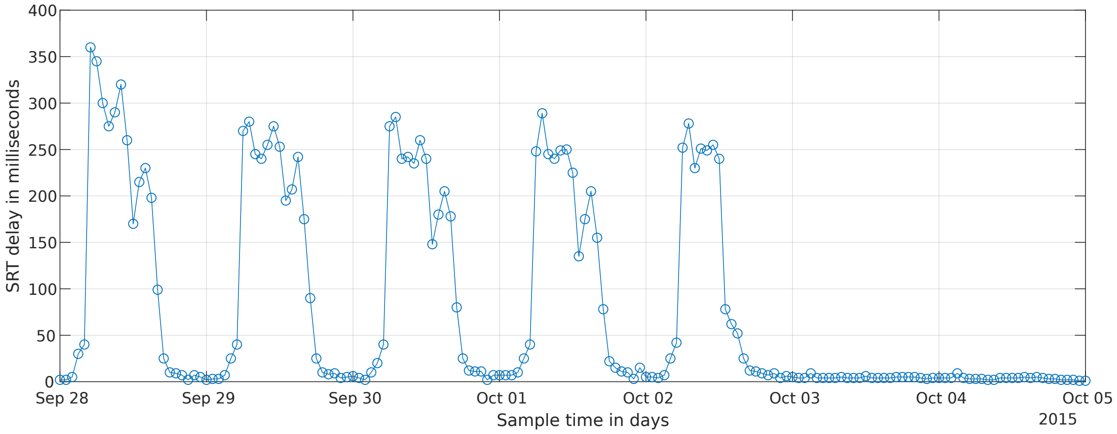

Fig. 2. SYN Response Time in milliseconds (one week, from Monday to Sunday) of a main server in a large company.

on historical load data, which is fed to an ARIMA model to estimate future load figures. In contrast, our approach uses actual data instead of previous load records.

Parameters such as the number of active connections in a given instant are used as an input to other load balancing systems [32]. This approach addresses the optimization of traffic distribution in a cluster of web servers. A method to balance traffic between servers is proposed, based on CPU load estimation. The load balancer algorithm uses two parameters from every server: the amount of exchanged traffic and the number of active connections. The CPU load is estimated as the ratio of the amount of the exchanged traffic over the server capacity. The rate at which those calculations and corresponding actions are made is optimized. Simulations show this rate is reduced almost three times. In our approach, we address accurate load measurement instead of a rough estimation, and using only the SRT of the connections. Note also that other approaches [33] are based on the number of SYN segments (actually, number of TCP connections vs. total capacity) received by the server. They do not provide an accurate measure of the server load because the load depends both on this arrival rate and on the service rate. This is usually unknown, and not necessarily related to the transmitted data.

Virtual Machine state transitions have also been used as a non-intrusive estimation of VM Load to classify huge amounts of VM instances when dealing with performance issues in a cloud [34]. An agentless technique for VM feature extraction is proposed. Hypervisor trace mining is used to extract static trace points. The states of every virtual CPU and virtual interrupt injection rate are used. Resource contention due to other VM and VM exit reasons are also included in the method. Therefore, without access to the VM itself, both coarse and fine grain workload data estimations are obtained. Once the workload is available, K-means clustering is used to isolate the VM that might have issues. Experimental data shows that, for instance, VM with CPU contention may be rapidly identified. Our approach avoids accessing the hypervisor and therefore is less intrusive.

In summary, those previous approaches lack generality and detail, and are highly suited to specific environments.

3. TCP SYN Response Time

In this section, we explain how the SRT is measured and its correlation with server load.

3.1. SRT measurement

The TCP protocol, requires a connection set-up before transmitting or receiving any useful application-layer data. The complete connection set-up involves three segments: SYN, from client to server, SYN+ACK, from server to client, and ACK, once again from client to server. The TCP protocol does not consider the connection established until all of them have been successfully received. This 3-way handshake, shown in Fig. 1, can be measured by using traffic probes [35] placed at a vantage point. In this way, this measurement is independent of where the client is located.

Any of the delays in the phases can be used as an estimation for Round Trip Time (RTT) [36], which in turn has been used to estimate network infrastructure issues [37]. However, it should be noted that in contrast to these previous works, we are not going to measure neither the RTT nor the network status. What it is presented here is an innovative approach that uses the SRT as an estimator of the server load. In this case, it is important to place the vantage point close to the server (e.g., between the server switch and the server itself), so network influence on measured the SRT is minimized.

3.2. SRT versus load correlation

As mentioned above, our previous work has shown that there is a strong correlation between distribution parameters and server load [23]. In Fig. 2 we can see the evolution in one week of the average SRT of a main server in a large company. The shape of the graphics suggests our guessing that the SRT and server load are related, at least at a daily level, given that the SRT follows the usual daily traffic pattern [38]. In different time scales, this relationship can be found even when using a

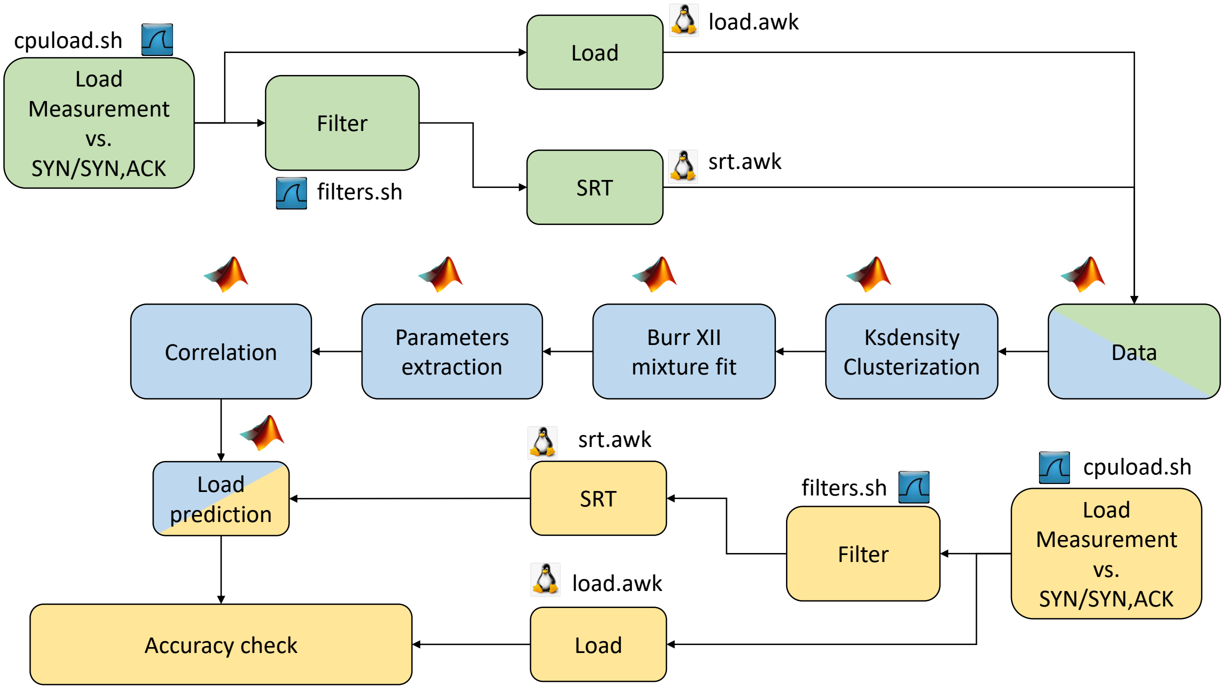

Fig. 3. Methodology used in this work. Data acquisition tasks are in green, server load estimation tasks in blue, and model validation in wheat.

single distribution to fit the SRT data. However, we show in this paper that better results can be achieved if Burr Type XII mixture is used.

4. Methodology

To find the proper correlation between the SRT and server load, we have defined the following methodology. With it, we can validate our approach, where actual data is used to feed the mathematical model, and results are compared to measured load. In this way, we can check that this method can be used in real premises.

Fig. 3 depicts the methodology workflow used in the present work. We can highlight three main phases:

1. *Data acquisition*. It is the starting point and the source to tune the load estimation algorithm. Enough measurements in a controlled environment (essentially known server load) allow a correlation fitting between the SRT statistics and load figures.

2. *Server load estimation model setup*. Once the correlation is established (model calibration), the model is fed with actual the SRT figures and estimate server load previously imposed on the target system.

3. *Model validation*. Estimations are finally compared to the actual loads to test the accuracy of the method.

In Fig. 3, *data acquisition* components are depicted in green, *server load estimation model setup* in blue and *model validation* in wheat. Icons in the figure are used to document the tools used in every step. We have implemented all these phases, making the code available in GitHub[1] for reproducibility. The next subsections describe in detail each phase.

4.1. Data acquisition

The data set used has been obtained by a custom environment meant to measure the SRT related to several loads on a server. Two computers composed this environment: one of them had the client role whilst the other one had the server role, and it worked as follows. The client was running on a Kali Linux virtual machine, while the server was running on an Ubuntu Linux operating system. Both of them directly connected through a physical Ethernet wire with IP address manually configured.

To measure the SRT times, TCP connection requests must be sent from one point to the other. A script located on the server called "`cpuload.sh`" (Fig. 3) has automatized this operation. It uses two different Python scripts "`Python-Server_200OK.py`" and "`Python-PoissonClient.py`", each one located on the server and the client, respectively, as their names indicate.

The function of the script "`Python-Server_200OK.py`" is to open a network port and keep it listening for upcoming TCP connections. Once a TCP connection arrives, the script accepts the connection request and sends back a "200 OK" answer. On the other hand, the script "`Python-PoissonClient.py`" keeps sending TCP connections with "GET" messages, through the same port used by the server, with a rate of 5 connections per

[1] `https://github.com/ARosu21/Load_estimation_by_SRT`

second, following a Poisson process. Note, however, that the SRT is independent of the application-level messages, which are only used to add some content to the connections.

As it has been said before, to automatize this procedure, the first thing done in the script “`cpuload.sh`” is loading the CPU of the server from 0% to 100% in 5% intervals using the following command: “`stress-ng -c 4 -l` *`%load`*”; the “`-c`” option allows selecting the number of cores to be loaded with the percentage of load selected with the “`-l`” option. Thereafter, the script “`Python-Server_2000K.py`” is launched, listening on the port number 2004. Once the server is running, the network traffic is captured by the program `tshark` using the next capture filter: “`tcp port 2004 and (tcp[tcpflags]&(tcp-syn)!=0)`”. With this filter, only TCP segments that go through the port number 2004 with any of the flags “SYN” or “SYN+ACK” are captured.

The next step is to launch the script located in the client. “`Python-PoissonClient.py`”. To do so, the server uses the SSH protocol to have access to the client and run the script. The final part is the measurement of the real load on the server. To obtain these measurements, the command “`mpstat 2`” is used, which measures several stats of the CPU such as “`%idle`” that indicates the amount of free CPU at that moment. In this case, it is measured every 2 seconds, so the current load can be determined by the subtraction: $load = 100 - \%idle$.

All this procedure is done for every percentage of load in the mentioned set, so it is repeated with all the 21 different percentages of load. Doing this for 15 minutes long and 5 times for every measure, the calibration data set is obtained, and after the calibration is concluded, the test data set is done in the same way but from 1 to 5 minutes long instead, and only once, to be later used for validation (see sect. 4.3).

Having captured the traffic and saved it in traces, it was time to filter it to get the SRT. The script called “`filters.sh`” (Fig. 3) does this process. To calculate the SRT, it is necessary to measure the time between “SYN” and “SYN+ACK” segments, previously sorted by arrival sequence, avoiding retransmissions or out-of-order segments. This is done with the following display filter in `tshark` that takes only “SYN+ACK” segments: “`tcp.flags==0x00000012 and not tcp.analysis.retransmission and not tcp.analysis.out_of_order`”, and taking the time since the previous frame in the TCP stream, “`tcp.time_delta`”. This filtered data is saved in a text file, which will be used to calculate the SRT with an AWK script “`srt.awk` (Fig. 3). In relation to the load, it is calculated with another AWK script “`load.awk` (Fig. 3) that does what has been said before. The subtraction of the `%idle` column and calculates the mean load for every load percentage setup.

4.2. Server load estimation model set up

We have tried, as a first approximation, the SRT distribution statistics like mean, median, etc. to predict the load, but the accuracy of the prediction is not accurate enough with such method. Bearing in mind the actual distributions of the experimental data (see Fig. 4), we concluded that better results may be obtained by fitting a distribution mixture. Statistics can be derived from the fitting, so more accurate estimations might be obtained. As per the Glivenko-Cantelli theorem, the more samples, the better results. However, there is a tradeoff between accuracy and estimation speed. Mixture fit can estimate the actual distribution, so not that many samples are needed to estimate distribution parameters. Therefore, we proceeded to fit a distribution mixture, so the measurement errors can be fixed (at least, partially). Experimental results support this assumption.

We used fifteen minutes of data as a good trade-off between accuracy and time to react to load changes. Data are shown in Fig. 3 as “Load Measurement vs. SYN/ACK”. Once we have the data, we cluster it using the K-density algorithm (Fig. 3), and then proceed with the Burr Type XII mixture fitting (details follow in section 5). Then, we calculate relevant statistics to predict the load. Once we find one of such statistics, we get the correlation fit, so we finally have a theoretical expression, which relates the SRT to server load. The result is depicted in Fig. 3 as “Load prediction” box. The output of this phase is a curve relating the SRT and load in a well-known environment, so it is actually a calibration process of the model. We were able to use many more samples than in an actual measurement, so fitting is excellent (see following sections).

4.3. Model validation

To validate the model, the SRT validation samples (different from calibration samples) are used to simulate a real data center environment. Depending on the actual setup and variability, the number of these new samples may vary significantly, but it is for sure much less than in the calibration phase. In our case, as stated before, we use 1-minute and 5-minutes timeframes to get the validation samples, vs. the 15-minutes timeframe used in the calibration phase. We have tested the validity of this approach in this way: We used the SRT data as an input to the “Load prediction” function (fitting curve) and then checked the result versus the actual server load. The result indicated that the estimation error when using the fitting curve was less than 5%.

The scripts used in this phase are the same as in *data acquisition* phase, as can be seen in Fig. 3.

4.4. Server Load Estimation Method

Based on the methodology provided above, this approach can be actually used in a data center to estimate the server load. The 3-step proposed method is as follows:

1. First, given the dependence of this estimation on the used hardware and software, it is necessary to obtain the correlation curves of the Burr Type XII mixture statistics with the load for the servers. This training phase would be similar to what has been shown previously, obtaining as a result a regression curve (second order seems enough) useful to estimate the server load.

2. After the training phase, the SRT samples have to be collected at the vantage point. This collection can be done non-intrusively by capturing with a probe the SYN and SYN+ACK segments of real TCP connections arriving to the server. Active SYN attempts may be used as explained,

or passively collect SYN segments from actual TCP connections.

3. Once the SRT samples have been collected, we can fit them to a Burr Type XII distribution mixture and obtain its statistics. Based on those values and on the regression curve obtained in the first step, we can finally estimate the server load.

5. Server Load Distribution Model

In this section, the mathematical model to fit the SRT data distribution is detailed. Although a single distribution might do the work, as shown in [23], we have observed that experimental data actually leads to a mixture. Using several loads in a server and measuring the SRT produces the histograms in Fig. 4, where it can be seen by inspection that the better approach is to consider it as a PDF mixture of heavy-tailed PDFs because, for all server loads from 0% to 100%, a heavy-tailed multimodal distribution is observed. Given this fact, we have developed a new mathematical approach to get the mixture components. The next subsections are organized as follows:

1. *Distribution Mixtures.* Distribution mixture notation and problem statement are defined.

2. *Burr Type XII Distributions.* The Burr Type XII distribution explicit expression is presented as an input for the following subsections.

3. *Mixture estimation using MLE.* Maximum Likelihood Estimation (MLE) has been used to estimate mixtures of Gaussian random variables successfully. Here the general approach is explained in detail, and its application to Burr Type XII distributions. The technique used to solve MLE in this approach is expectation maximization (EM) which is also presented in this subsection.

4. *Burr Mixture EM Second Term Optimization.* As a difference with the Gaussian approach, one of the EM terms should be maximized by using numerical analysis. This subsection presents the method to get that maximum.

5. *Burr mixture fitting algorithm.* Finally, the whole algorithm is summarized, and mathematical details are discussed.

5.1. Distribution Mixtures

A mixture can be defined as a linear combination of a finite or infinite number of PDFs, called components. In the case that all the PDFs are of the same type and the number of components is finite, the general expression is shown in Eq. 1:

$$\begin{cases} p\left(x\middle|\bar{\theta}\right) = \sum_{l=1}^{M} \alpha_l \, p_l\left(x|\theta_l\right) \\ \sum_{l=1}^{M} \alpha_l = 1 \end{cases} \tag{1}$$

where α_l are the relative weights of every component of the mixture and $\bar{\theta}$ is the set of parameters that define each PDF. The objective is to find both sets of parameters to estimate the mixture PDF. Several techniques have been proposed to achieve this goal. The machine learning community often uses clustering to separate the data into clusters, then fit the PDF to every cluster and combine them to get the overall mixture [39]. This method works quite well when the mixture components are separated enough one from another, but if the components are "wedged", extra process is needed to improve accuracy. Another method is based on the Bayes approach, which can be seen in [40], but we have decided to use MLE, which has proven very efficient when dealing with Gaussian distributions [41].

5.2. Burr Type XII distributions

It makes sense that fitting quality is key to estimating server load accurately. One alternative is to use MLE techniques to improve the fitting. As we will see, MLE applications are far more efficient when PDF expressions are available, which is not possible when using, for instance, α-stable distributions, which do not have a closed expression for the PDF or CDF. As an alternative, Burr Type XII distributions [27] have an explicit expression for both PDF and CDF functions, as shown in Eq. 2:

$$\begin{cases} f\left(x|a,c,k\right) = \frac{c\,k}{a}\left(\frac{x}{a}\right)^{c-1}\left[1+\left(\frac{x}{a}\right)^{c}\right]^{-(k+1)} \\ F\left(x|a,c,k\right) = 1 - \left[1+\left(\frac{x}{a}\right)^{c}\right]^{-k} \end{cases} \tag{2}$$

where a is the scale parameter and c and k are the shape parameters. Furthermore, Burr Type XII distribution has been used to successfully model several experimental data sets like house income in the US or loss expectation [28]. They are suitable to fit heavy-tailed distributions and can be manipulated explicitly, so they have proven applicable to a number of heavy-tailed models [29]. In this paper, we will take advantage of the explicit PDF expression to apply MLE to find out the mixture components of an SRT experimental histogram.

5.3. Mixture estimation using MLE

As it has been previously mentioned, MLE has been applied to Gaussian mixture estimation successfully due to the explicit PDF expression [42]. The idea is to use the explicit expressions of the PDF functions and then, use the partial derivatives to find the maximum (EM). Other proposed methods for a defined number of components are based on Weibull distribution properties [43] or on Bayes' rule [44, 45]. Actually, MLE has been used successfully to estimate the three Burr Type XII parameters for a single component [46]. We will follow the method explained in [41] for Gaussian distribution, but using Burr Type XII instead, which in our experience provides a better fit. Using the mixture definition in Eq. 1, the incomplete-data log-likelihood expression for this density for the data X is given in Eq. 3:

$$\begin{aligned} \log\left(\mathcal{L}\left(\bar{\theta}\middle|\mathcal{X}\right)\right) &= \log\prod_{i=1}^{N} p\left(x_i\middle|\bar{\theta}\right) \\ &= \sum_{i=1}^{N}\log\left(\sum_{l=1}^{M}\alpha_l\, p_l\left(x_i|\theta_l\right)\right) \end{aligned} \tag{3}$$

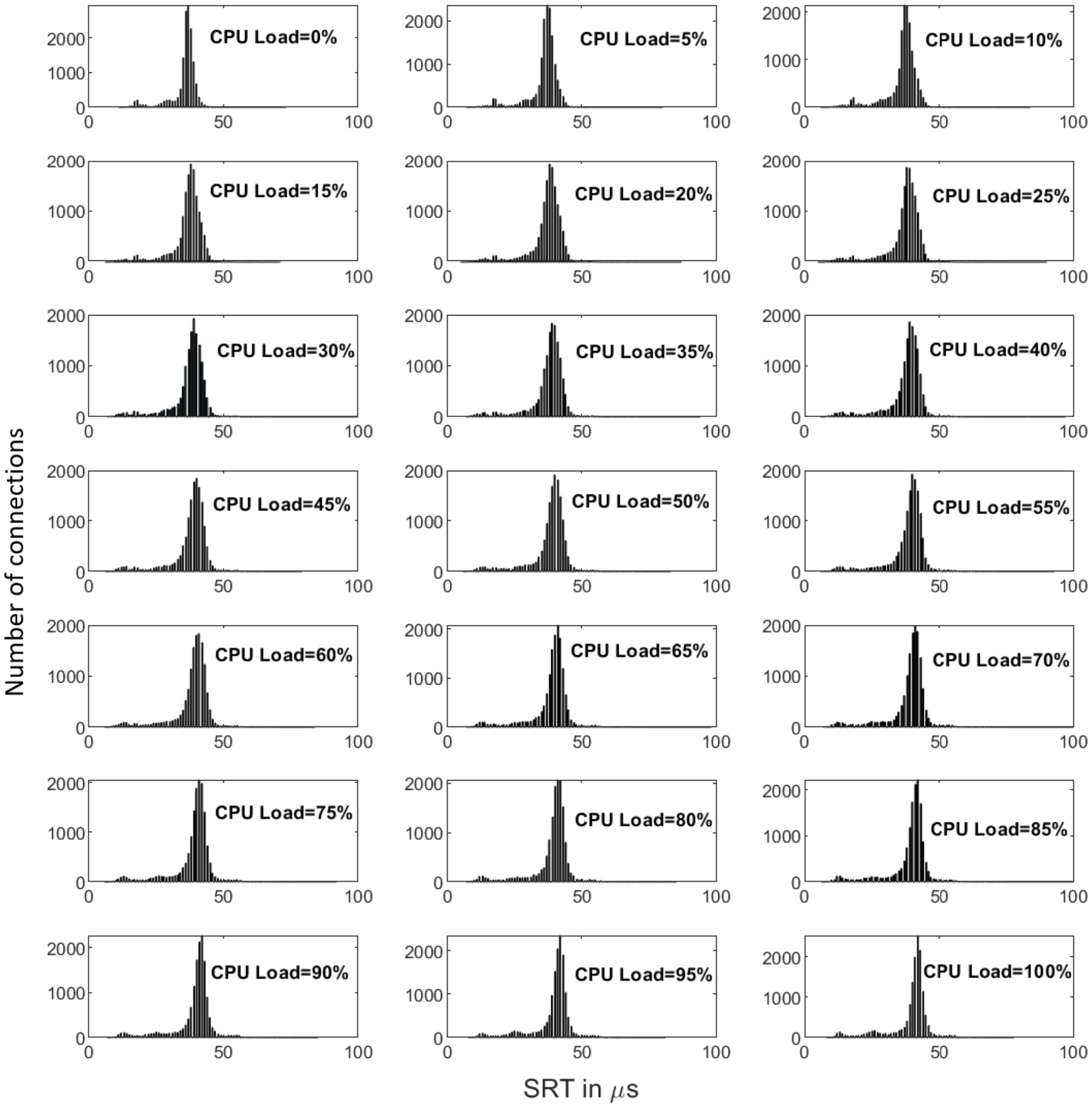

Fig. 4. SYN Response Time histogram for different CPU loads, ranging from 0% (top left) to 100% (bottom right).

To optimize the log-likelihood expression, we post the existence of a dataset $\mathcal{Y} = \{y_i\}_{i=1}^{N}$, which indicates for every sample which component it belongs to. That is, $y_i \in 1, ..., M$ and $y_i = k$ if the i^{th} sample is generated by the k mixture component.

If we consider the $\mathcal{Y}$ set as a random vector of unobserved data items, we can apply MLE. Likelihood expression becomes now as shown in Eq. 4:

$$\begin{aligned} \log\left(\mathcal{L}\left(\bar{\theta}\middle|X, Y\right)\right) &= \log\left(P(X, Y|\bar{\theta})\right) \\ &= \sum_{i=1}^{N} \log\left(P(x_i|y_i)\, P(y)\right) \\ &= \sum_{i=1}^{N} \log\left(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})\right) \end{aligned} \tag{4}$$

We need to start with an initial guess at the mixture parameters $\bar{\theta}^g = (\alpha_1^g, ..., \alpha_M^g, \theta_1^g, ..., \theta_M^g)$. With this guess, we can compute every mixture component $p_l(x_i, \theta_l^g)$. To obtain the unobserved data distribution, we can use Bayes' rule as shown in Eq. 5:

$$\begin{aligned} p(y_i|x_i, \bar{\theta}^g) &= \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{p(x_i|\bar{\theta}^g)} \\ &= \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{\sum_{k=1}^{M} \alpha_k^g p_k(x_i|\theta_k^g)} \end{aligned} \tag{5}$$

Therefore, the probability of an instance of the unobserved data $\bar{y}$ is $p(\bar{y}|X,\ \bar{\theta}^g) = \prod_{1=1}^{N} p(y_i|x_i,\ \bar{\theta}^g)$. To maximize the likelihood, we may use the expectation maximization (EM) algorithm. We define the Q function in Eq. 6, which is the first step (called E) where we calculate the expectation:

$$\begin{aligned} Q(\bar{\theta},\ \bar{\theta}^{(i-1)}) &= E\left[\log\ p(X, Y|\bar{\theta})|\ X, \bar{\theta}^{(i-1)}\ \right] \\ &= \int_{y\in\Upsilon} \log\ p(X, \bar{y}|\bar{\theta}^{(i)}) f(\bar{y}|X, \bar{\theta}^{(i-1)}) d\bar{y} \end{aligned} \tag{6}$$

where X and $\bar{\theta}^{(i)}$ are constants (data and previous parameter estimation) and $\bar{y}$ is an instance of the unobserved data.

The second step in the EM algorithm (called M) is to maximize Q :

$$\bar{\theta}^{(i-1)} = \arg\max_{\bar{\theta}} Q\left(\bar{\theta}, \bar{\theta}^{(i-1)}\right) \tag{7}$$

In the mixture fitting case, the Q function is as in Eq. 8:

$$Q(\bar{\theta},\ \bar{\theta}^{(i-1)}) = \sum_{\bar{y}\in\Upsilon} \sum_{i=1}^{N} \log\left(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})\right) \prod_{j=1}^{N} p(y_j|x_j, \bar{\theta}^g\ ) \tag{8}$$

This expression can be transformed into a sum of two separate expressions, which can be independently optimized as seen in Eq. 9:

$$\begin{aligned} Q(\bar{\theta},\ \bar{\theta}^{(i-1)}) &= \sum_{l=1}^{M} \sum_{i=1}^{N} \log\left(\alpha_l) p(l|x_i, \bar{\theta}^g)\right) + \\ &\sum_{l=1}^{M} \sum_{i=1}^{N} \log\left(p_l(x_i|\theta_{y_i})\right) p(l|x_i, \bar{\theta}^g) \end{aligned} \tag{9}$$

The first term can be optimized using Lagrange multipliers. Eq. 10 shows the expression for $\alpha_l, l = 1...M$.

$$\alpha_l = \frac{1}{N} \sum_{i=1}^{N} p(l|x_i, \bar{\theta}^g) \tag{10}$$

In the case of Gaussian mixtures, the second term in Eq. 9 can be optimized, finding values for the partial derivatives to vanish. However, when dealing with Burr Type XII distributions, we have used a numerical approach.

5.4. Burr Mixture EM Second Term Optimization

If we include the Burr Type XII PDF definition into the EM second term, we get the function to be maximized shown in Eq. 11:

$$\begin{aligned} &\sum_{l=1}^{M} \sum_{i=1}^{N} \log\left(p_l(x_i|\theta_{y_i})\right) p(l|x_i, \bar{\theta}^g) = \\ &\sum_{l=1}^{M} \sum_{i=1}^{N} \log\left(\frac{c_l\, k_l}{a_l} \left(\frac{x_i}{a_l}\right)^{c_l-1} \left[1 + \left(\frac{x_i}{a_l}\right)^{c_l}\right]^{-(k_l+1)}\right) \cdot \\ &p(l|x_i, \bar{\theta}^g) = \\ &\sum_{l=1}^{M} \sum_{i=1}^{N} \{\log c_l + \log k_l - \log a_l + (c_l - 1) \left[\log x_i - \log a_l\right] \\ &- (k_l + 1) \log\left[1 + \left(\frac{x_i}{a_l}\right)^{c_l}\right]\} p\left(l\middle|x_i, \bar{\theta}^g\right) \end{aligned} \tag{11}$$

As every mixture component is independent of the rest, we can optimize every function f_l with $l = 1...M$ described in Eq. 12:

$$\begin{aligned} f_l(a_l, c_l, k_l) = \sum_{i=1}^{N} \{&\log c_l + \log k_l - \log a_l + \\ &(c_l - 1) \left[\log x_i - \log a_l\right] \\ &- (k_l + 1) \log\left[1 + \left(\frac{x_i}{a_l}\right)^{c_l}\right]\} p\left(l\middle|x_i, \bar{\theta}^g\right) \end{aligned} \tag{12}$$

The partial derivatives of f_l are shown in Eq. 13:

$$\begin{cases} \frac{\partial f_l}{\partial a_l} = \sum_{i=1}^{N} \left[-\frac{c_l}{a_l} + (k_l + 1)\, c_l {x_i}^{c_l} \frac{a_l^{(c_l+1)}}{1+\left(\frac{x_i}{a_l}\right)^{c_l}}\right] \\ \qquad \cdot\, p\left(l\middle|x_i, \bar{\theta}^g\right) \\ \frac{\partial f_l}{\partial c_l} = \sum_{i=1}^{N} \left[\frac{1}{c_l} + \log\left(\frac{x_i}{a_l}\right) - (k_l + 1) \frac{\log\left(\frac{x_i}{a_l}\right)\left(\frac{x_i}{a_l}\right)^{c_l}}{1+\left(\frac{x_i}{a_l}\right)^{c_l}}\right] \\ \qquad \cdot\, p\left(l\middle|x_i, \bar{\theta}^g\right) \\ \frac{\partial f_l}{\partial k_l} = \sum_{i=1}^{N} \left[\frac{1}{k_l} -\ \log\left(1 + \left[\frac{x_i}{a_l}\right]^{c_l}\right)\right] \cdot p\left(l\middle|x_i, \bar{\theta}^g\right) \end{cases} \tag{13}$$

To find the maximum of f_l, we can numerically find the (a_l, c_l, k_l) values where Eq. 13 vanishes. Actually, we can find an expression of k_l when $\frac{\partial f_l}{\partial k_l} = 0$ as a function of (a_l, c_l) values. Eq. 14 shows the expression for $k_l(a_l, c_l)$

$$k_l\ (a_l, c_l) = \frac{\sum_{i=1}^{N} p\left(l\middle|x_i, \bar{\theta}^g\right)}{\sum_{i=1}^{N} \log\left[1 + \left(\frac{x_i}{a_l}\right)^{c_l}\right] p\left(l\middle|x_i, \bar{\theta}^g\right)} \tag{14}$$

We can use this result to reduce the three dimensions search of the maximum to two dimensions, as k_l can be considered a function of the other two parameters. The function g_l to be maximized is shown in Eq. 15:

$$\begin{aligned} g_l(a_l, c_l) \;=\; & f_l(a_l, c_l, k_l(a_l, c_l)) \;= \\ & \sum_{l=1}^{M} \sum_{i=1}^{N} \{\log c_l \\ & + \log\left(\frac{\sum_{i=1}^{N} p\left(l \middle| x_i, \bar{\theta}^g\right)}{\sum_{i=1}^{N} \log\left[1 + \left(\frac{x_i}{a_l}\right)^{c_l}\right] p\left(l \middle| x_i, \bar{\theta}^g\right)} \right) - \\ & \log a_l + (c_l - 1)\left(\log \frac{x_i}{a_l}\right) \\ & \left[\left(\frac{\sum_{i=1}^{N} p\left(l \middle| x_i, \bar{\theta}^g\right)}{\sum_{i=1}^{N} \log\left[1 + \left(\frac{x_i}{a_l}\right)^{c_l}\right] p\left(l \middle| x_i, \bar{\theta}^g\right)} \right) + 1\right] \\ & \log\left[1 + \left(\frac{x_i}{a_l}\right)^{c_l}\right]\} \cdot \\ & p\left(l \middle| x_i, \bar{\theta}^g\right) \end{aligned} \tag{15}$$

To find the maximum using the Monte Carlo method, we can proceed as follows for every random sample. We use a_l and c_l in $\bar{\theta}^g$ values as the previous guess. The Monte Carlo maximization algorithm follows these steps:

- Generate (a_l, c_l) random values within interval around previous guess values.
- Calculate $g_l(a_l, c_l)$ using Eq. 15.
- Follow maximum slope using $(\frac{\partial f_l}{\partial a_l}, \frac{\partial f_l}{\partial c_l})$ direction with Eq. 13 and calculate $g_l(a_l, c_l)$ again using Eq. 15 in every step.
- Repeat until there is not increasing in the value of $g_l(a_l, c_l)$.
- Make (a_l, c_l) part of $\bar{\theta}^g$ as guess values for the next EM iteration.

We can remark several points regarding the Monte Carlo method:

- The interval for the random values depends on the accuracy of the initial estimations. Hopefully, we start with a pretty good estimation, so we may use a small interval for the (a_l, c_l) values.
- The gradient we are using is the f_l gradient, not the g_l one. As we are imposing $\frac{\partial f_l}{\partial k_l} = 0$, this is not a big concern. We will see that results support this assumption.
- The Monte Carlo Method requires going through the described steps several times, depending on the actual problem. We have found that the required number of samples is in the order of tenths.

5.5. Burr mixture fitting algorithm

Once we have the method to optimize the second term of $Q(\bar{\theta},\ \bar{\theta}^{(i-1)})$ in Eq. 9 and the α_l values with Eq. 10, we can iterate EM until fit the distribution. The overall EM mixture-fitting method with Burr Type XII distributions is as follows:

- Find mixture maxima as initial cluster centroids.
- Use k-means with Euclidean distance to define clusters of samples.
- Fit Burr Type XII distributions to every cluster of samples. These fits are the initial components of the mixture.
- Estimate α_l for every component.
- Define initial $\bar{\theta}^g$ using Burr Type XII parameters and α_l values.
- Iterate:
  - Using Monte Carlo, estimate new $\{(a_l, c_l, k_l)\}, l = 1...M$.
  - Calculate α_l using Eq. 10.
  - Update $\bar{\theta}^g$ for next iteration.

As previously, some discussion on the presented method is worth:

- EM guarantees that with every iteration expectation the result is improved, even if the absolute maximum is not found. This allows us to define a stop criterion for the iterations, like a distance measure (Kolmogorov-Smirnoff for instance [47]) minimum threshold.
- In every iteration, the two terms of Eq. 9 are optimized. It may improve the convergence of the algorithm if we use one of the optimizations as input for the other. However, we have chosen not to update $\bar{\theta}^g$ until the end of the iteration, as not significant speed improvement is obtained.
- Obviously, the closer the initial estimation to the solution, the better for the algorithm to converge to the solution. We have noted that maxima estimation is probably the most critical step to estimate the initial components. Using information from the problem can help us. As we have seen in Fig. 4, there are four components in the mixture, and the third one is by far the most important. We first find the absolute maximum of the mixture, and then impose two maxima to the left and one to the right. We try to get the maxima the most possible equispaced. This pattern depends on the server architecture and should be defined case by case.
- Once the samples are classified in clusters, we need to estimate initial α_l values. We use the maxima from the initial estimation and take advantage of the Burr Type XII distribution. For every component, we calculate the PDF peak

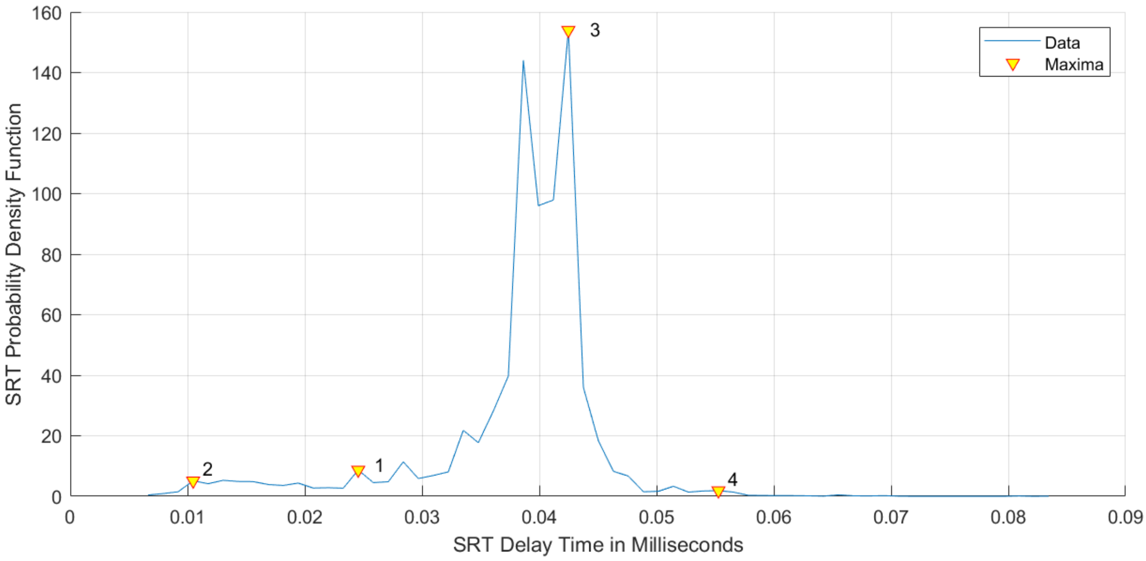

Fig. 5. SRT histogram maxima locations.

and calculate the ratio to the corresponding maxima. Then, we normalize those ratios and use them for the α_l values to make sure the sum of all of them is one to comply with the second axiom of probability.

By using the presented method, we have been able to fit the mixture using Burr Type XII distributions in a much better way than with α-stables [23], which do not have a closed form to deal with. Results are presented in the following section.

6. Experimental results

Once we have described the mathematical model, we have applied it to the experimental data, to obtain the distribution mixture. The objective is later to find the curve that better models the relation between the SRT and the server load.

6.1. Burr Type XII fit

To test the presented algorithm, we have used as a target server to measure an Intel i5 430M, 2 cores, frequency 2.26 - 2.53 GHz, 3 MB cache 4 GB RAM DDR3 at 1067 MHz with HDD disk of 500 GB. The operating system is Ubuntu 16.04 LTS, which reports four cores (two physical and two logical). Data has been acquired using the methods described in section 4.1.

Once we have the SRT and load data, the first step is to localize mixture maxima. As mentioned, we use the usual criterion for the number of bins (square root of the number of samples) and then apply the maxima finding procedure already explained, bearing in mind the width of the bins. In Fig. 5 we can see the result of a problematic case. This is especially interesting because every so often, we do not have an “isolated” maximum for the main mixture component. This implies that using a blind approach to extract just the maxima does not work because we would get two quite close maxima in this case and miss some other maximum either on the right or on the left. In fact, any of the two maxima in the center of the figure could work because we have MLE to improve the fit. Then, we choose any of them and forget the other, while finding maxima separated enough from the one already chosen. We have checked that this approach using separated maxima gets far better results than just a maxima naive finding.

The next step is to use maxima locations as initial locations for the k-means clustering. In Fig. 6 we can see the four clusters of samples, the corresponding histograms without normalization (each one is a PDF itself) and the initial Burr Type XII distribution fit for every component.

We should make a remark here regarding Burr Type XII fit to every cluster histogram. Sometimes (very few, actually) the best fitting PDF is a limiting case of Burr Type XII like Weibull distribution [46]. In this case, some parameters have infinite values, the Burr Type XII PDF expression cannot be used directly. We have decided to estimate a “standard” Burr Type XII PDF with $c = 3$ and $k = 1$ parameters as it has “bell shape” like the components. To approach the “not fittable” distribution, the Burr Type XII a parameter is assigned to the corresponding cluster centroid location. Sometimes, resampling of fitting curves is necessary. This heuristic approach has proven to be a valid assumption for the case of study. Once the α_l parameters are estimated as described, we can see in Fig. 7 the initial Burr Type XII mixture estimation. The final step is to iterate EM to improve the fitting. In this case, we have used 100 random samples for the Monte Carlo algorithm and 20 iterations. The random variable interval is $[-30\%, +30\%]$ of the guessed values ($\bar{\theta}^g$) decreasing with every iteration. In Fig. 8, the resulting Burr Type XII mixture estimation is depicted.

6.2. Server Load correlation with the SRT

We have repeated the previous method for mixture fitting for several loads, between 5% and 100%. After several attempts us-

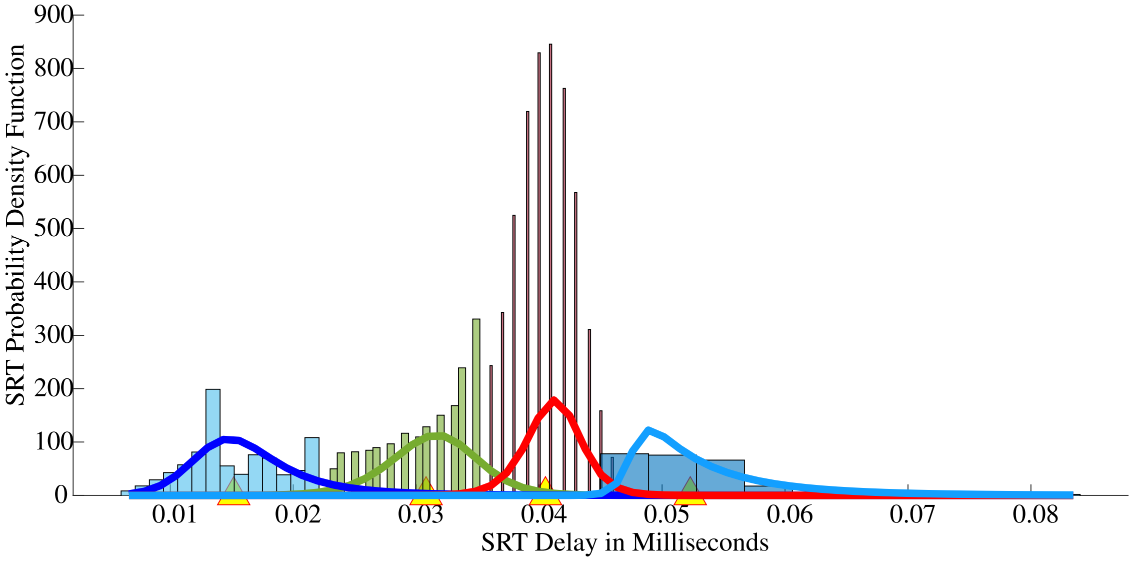

Fig. 6. Clustering without normalization. Different color lines identify the fitting of each mixture component.

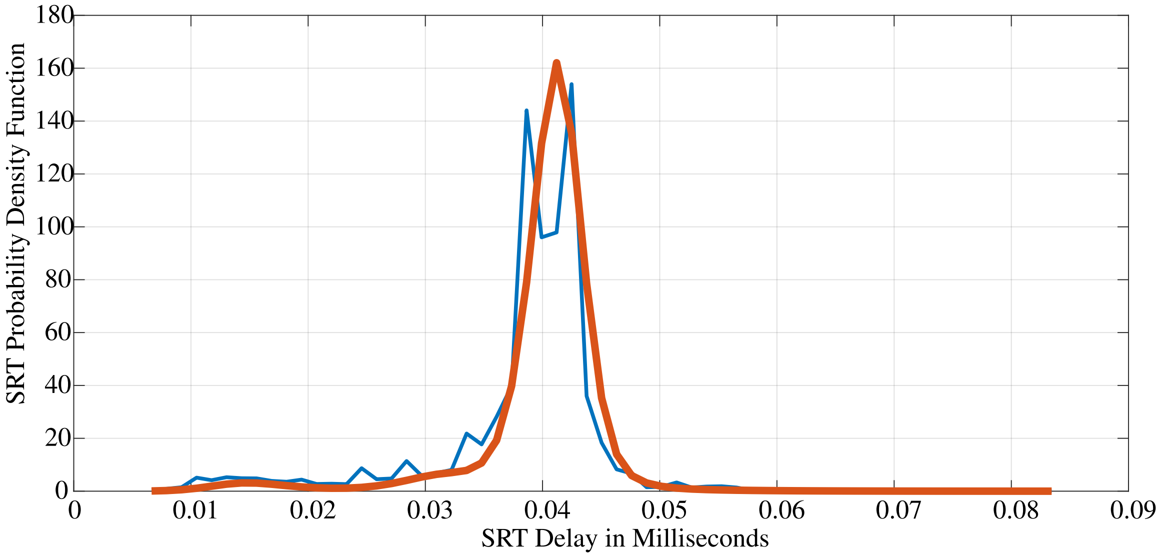

Fig. 7. Initial Burr Type XII measured data (blue line) and fitting (red line).

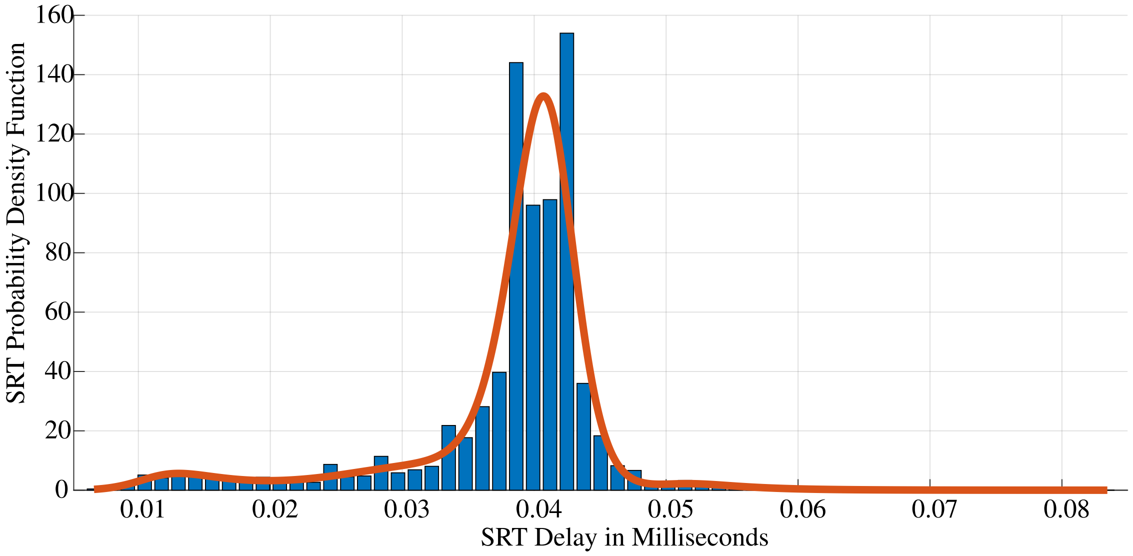

Fig. 8. Burr Type XII mixture measurement (blue bars) and fitting after EM (red line).

ing mean, mode and other statistics, we finally decided that the best correlation between server load and the SRT statistic is the 50th percentile (actually, the median) of the estimated mixture, but far from satisfactory. Experimental results are presented in Fig. 9

Median captures somehow the location of the data (essentially, delays) which hopefully has to do with the server load. The median is generally considered more efficient than the mean when addressing heavy-tailed distributions or mixtures [48]. However, although other statistics are even worse, empirical median does not provide the accuracy obtained with Burr Type XII mixtures.

In Fig. 10 we can see a correlation between the percentile and the server load. Markers represent measurements and the continuous line is the second order polynomial adjust, $y = 3.6x^2 + 27.6x + 44.1$. Dash lines indicate the 95% confidence interval. For the polynomial fit, we have $\mu = 0.039286$ and $\sigma = 0.001161$. This is an excellent correlation and that is why we propose it as the right method to estimate server load without intrusion into the system.

To assess how the method performs, we estimate the error of the estimated load. For this, as proposed in [9], we use the Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE), defined below in equations 16 and 17, respectively.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |e(i)| \tag{16}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \frac{|e(i)|}{x(i)} \cdot 100\% \tag{17}$$

where $e(i) = \hat{x}(i) - x(i)$, $\hat{x}(i)$ is the estimated load value of the i^{th} measurement based on the SRT, and $x(i)$ is the actual mean load value provided by `mpstat` as ground truth. Note that, in this case, we provide the loads as percentages, so MAE will be a percentage as well. Anyway, we also use MAPE to identify if there are large errors when the load is low, which could be hidden in MAE. Additionally, apart from MAE and MAPE, we obtain the maximum absolute and percentage errors to know the error upper bound. Based on the values shown in Fig. 10, we have obtained the following results for our experiment:

- Average absolute error: 0.72%.
- Maximum absolute error: 2.01%.
- Average absolute percentage error: 2.01%.
- Maximum absolute percentage error: 5.89%.

As shown, our results outperform other load estimation methods [9]. There, MAPE was between 4.31% and 7.22% for ClarkNet, and between 3.09% and 5.16% for NASA. Thus, our approach provides a lower MAE and MAPE, proving its usefulness regarding leading related techniques.

7. Discussion

The described method supposes that the TCP SYN responses are generated by the operating system in the server to correctly estimate the server load. However, there are many network cards with TCP offloading features, which could make this method unfeasible. Nevertheless, most TCP offloading implementations only do partial offloading. They take the TCP control once the connection has been established by the operating system, for tasks such as checksum calculation or segmentation/reassembly [49]. In these cases, the proposed method is still valid. The operating system is affected by the server load during the TCP handshake, which is when the measurements are taken. Thus, other connection-oriented protocols with an initial handshake (e.g., SCTP or QUIC) could also be used to estimate the server load.

The proposed method requires calibration before actual estimations. This can be done before the server is moved to production as a previous task for every system to be controlled.

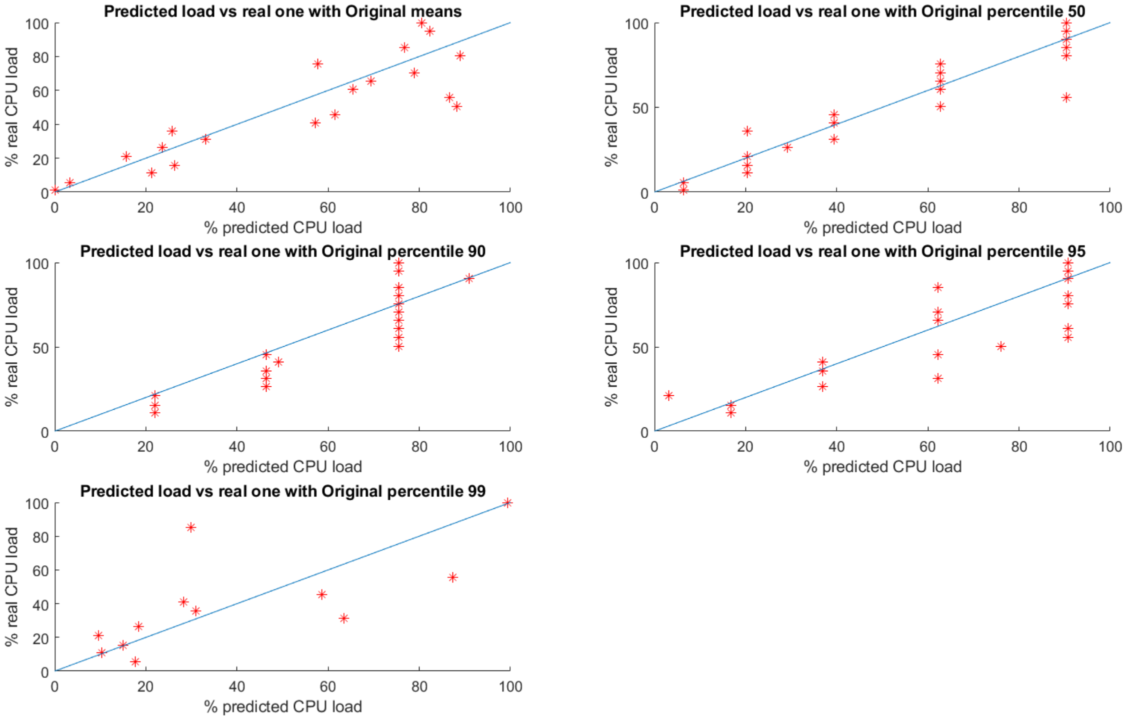

Fig. 9. Experimental statistic parameters vs. server load.

For already moved to production servers, low usage periods like non-labor hours or bank holidays can be used for calibrations. In case none of these alternatives is feasible, an actual twin system fully installed with all the hardware and software in production can be used for calibration. In cluster environments with many identical systems in production, it should not be an issue to calibrate the method one way or another.

The presented results are based on measurements at a vantage point close to the server whose load is to be measured. This helps to avoid other traffic that could interfere with the measurements, as shown in Fig. 1. In this way, a load balancer in front of a server cluster can do these measurements during its common operation. The farther the vantage point is from the server, the worse the estimation of the SRT, so it is important to deal with the vantage point location. However, it might be impossible to directly connect the probe to the system to be measured. It is not obvious what the effect of network equipment in between both systems is. Preliminary data shows that the mixture components may “compress” due to the network electronics queuing (mainly switches and routers) and a single distribution may be enough. In that case, accuracy is a concern, as in-the-middle equipment may jeopardize measurements because head-of-line blocking and buffering introduce noise to the measurement.

Regarding the complexity of the measurement method, it requires capturing TCP SYN segment times, fitting the Burr Type XII mixture distribution and taking its median value to translate it to the estimated load with the obtained curve in the calibration process. Measuring the SRT can be done with a script that captures the traffic with a capture filter to take only those segments with the SYN flag activated. There is no need to store the network traffic, just the SRT values for each connection. Once the distribution fitting is calculated, the actual load estimation calculation is straightforward. For instance, estimations in an Intel® Core™ i7-8565U CPU @ 1.80GHz, CPU with four cores and 8.0 GB of RAM, provide the following average figures:

- Maxima finding time: 140.63 ms.
- Clustering time: 375.00 ms.
- Initial fit time: 11.17 s.
- EM time per iteration: 13.74 s.
- Quadratic function evaluation: 8.75 μs.

Consider that those figures are obtained using an ordinary personal computer. The algorithm is tested in Matlab version 2020. Therefore, no optimization has been done so far. Even so, the elapsed time for the load estimation is about one minute (four EM iterations are usually enough). As per the SRT measurement period is five minutes, even in current implementation, the algorithm is suitable for actual measurement. Thus, these computing times strengthen the applicability of our approach.

Finally, it is also important to deal with the security implications of these results. Cybercriminals could use this technique to know when the servers are more loaded and attack them at that time, using fewer requests to cause a denial of service (DoS). In this case, attackers are usually far from the server. Consequently, the SRT estimation will be worse than the es-

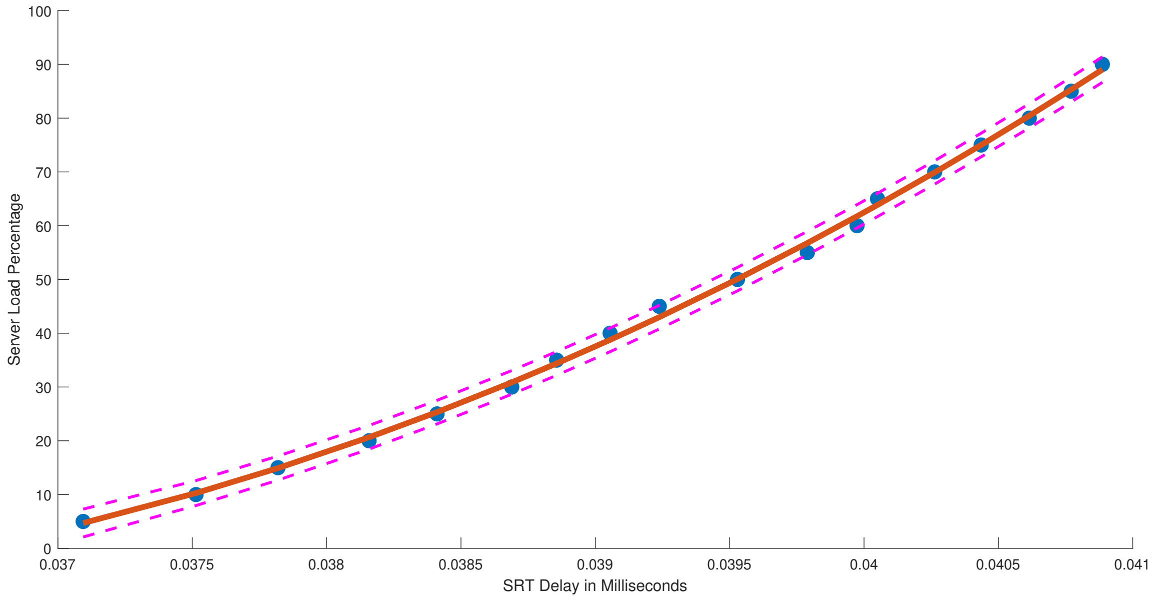

Fig. 10. SRT median in ms vs. server load with 95% confidence interval (dashed lines).

timation from our vantage point, next to the server. Here, to reduce the distance effect, network RTT may also be measured (e.g., with a ping). Then, the corrected SRT can be derived by subtracting the network RTT from the "raw" SRT measure. A similar approach has been proposed for different hops in [37]. This research idea will be further investigated.

8. Conclusions

In this paper, a novel approach has been presented, based on the SRT time, to estimate server load without measuring it internally. This approach is very valuable to identify bottlenecks causing problems in distributed systems, as well as to balance the load in server clusters or in the cloud.

For this, we measured the time from the SYN to the SYN+ACK TCP segments at the server side. We have identified that the SRT varies along the day in servers, following their workload. the SRT is distributed with a heavy tail, which is well modeled by a Burr Type XII mixture. Finally, we have found that server load is correlated with 50^{th} percentile of the mixture, but this is not useful at all. Better results are obtained by Burr Type XII mixture fitting.

Based on these results, an estimation method has been defined, which follows an initial training phase, where the server load is characterized. Then, a monitoring phase where the SRT samples are taken to find the load distribution of the server, based on the obtained Burr Type XII mixture statistics. Apart from the TCP connections, this new method does not take any assumptions about the server, which is considered a black box. Thus, it can be applied to almost any existing load balancing or scheduling algorithm.

This approach opens new research lines to be addressed. For instance, it is interesting to study how accurate the model is when there is network equipment between the server and the vantage point. This may be quite complex, as network topology may cancel the server load information reflected in the SRT distribution. On the other hand, it is also important to develop automatic ways for server calibration, based, for instance, on its operating system and hardware specifications, to ease the adoption of this load estimation method.

Acknowledgment

This work was supported in part by the Spanish Research Agency under the project AgileMon (AEI PID2019-104451RB-C21).

References

[1] R. Alshahrani, H. Peyravi, "Cluster Load Estimation for Stateless Schedulers," in Proc. IEEE 17th International Symposium on Network Computing and Applications (NCA), November 2018.

[2] Fouzi Semchedine, Louiza Bouallouche-Medjkoune, Djamil Aïssani, "Task assignment policies in distributed server systems: A survey," *Journal of Network and Computer Applications*, Volume 34, Issue 4, 2011, Pages 1123-1130.

[3] Ulrich Lampe, Markus Kieselmann, André Miede, Sebastian Zöller, Ralf Steinmetz, "On the Accuracy of Time Measurements in Virtual Machines," In: Rong Chang, Ephraim Feig, Alan Sussman, Liana L. Fong: Proceedings of the 6th International Conference on Cloud Computing (CLOUD 2013), Pages 103-104, Institute of Electrical and Electronics Engineers (IEEE), June 2013. ISBN 978-0-7685-5028-2.

[4] Avnish Thakur, Major Singh Goraya, "A taxonomic survey on load balancing in cloud," Journal of Network and Computer Applications, Volume 98, 2017, Pages 43-57, ISSN 1084-8045.

[5] Mosab Hamdan, Entisar Hassan, Ahmed Abdelaziz, Abdallah Elhigazi, Bushra Mohammed, Suleman Khan, Athanasios V. Vasilakos, M.N. Marsono, "A comprehensive survey of load balancing techniques in software-defined network," Journal of Network and Computer Applications, Volume 174, 2021, Article 102856, ISSN 1084-8045.

[6] Sumair Khan, Babar Nazir, Iftikhar Ahmed Khan, Shahaboddin Shamshirband, Anthony T. Chronopoulos, "Load balancing in grid computing: Taxonomy, trends and opportunities," Journal of Network and Computer Applications, Volume 88, 2017, Pages 99-111, ISSN 1084-8045.

[7] Xiao Qin, "Performance comparisons of load balancing algorithms for I/O-intensive workloads on clusters," Journal of Network and Computer Applications, Volume 31, Issue 1, 2008, Pages 32-46, ISSN 1084-8045.

[8] Veerabhadra R. Chandakanna, Valli K. Vatsavayi, "A sliding window based Self-Learning and Adaptive Load Balancer," Journal of Network and Computer Applications, Volume 56, 2015, Pages 188-205, ISSN 1084-8045.

[9] Jingjing Guo, Chunlin Li, Yi Chen, Youlong Luo, "On-demand resource provision based on load estimation and service expenditure in edge cloud environment," Journal of Network and Computer Applications, Volume 151, 2020, Article 102506, ISSN 1084-8045.

[10] Deepak Kumar Patel, Devashree Tripathy, C.R. Tripathy, "Survey of load balancing techniques for Grid," Journal of Network and Computer Applications, Volume 65, 2016, Pages 103-119, ISSN 1084-8045.

[11] Y. M. Teo, R. Ayani, "Comparison of Load Balancing Strategies on Cluster-based Web Servers," *SIMULATION*, Vol. 77, No. 5-6, Pages 185-195, 2001.

[12] K. Shen, T. Yang, L. Chu, "Cluster load balancing for fine-grain network services," in Proc. 16th International Parallel and Distributed Processing Symposium, Ft. Lauderdale, FL, 2002.

[13] S. Taylor, J. Metzler, "Eliminating the mean time to innocence," *Network World*, Mar 10, 2009.

[14] Víctor Fernández, Juan Manuel Orduña, Pedro Morillo, "Server implementations for improving the performance of CAR systems based on mobile phones," Journal of Network and Computer Applications, Volume 44, 2014, Pages 72-82, ISSN 1084-8045.

[15] Ashish Nadkarni, Eric Sheppard, Brad Casemore "Data Center Energy and Carbon Emission Reductions Through Compute, Storage, and Networking Virtualization," *IDC Executive Summary*, September 2017.

[16] Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab. Hamid, Muhammad Shiraz, Abdullah Yousafzai, Feng Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *Journal of Network and Computer Applications* Volume 52, 2015, Pages 11-25.

[17] M. Conti, Q. Q. Li, A. Maragno, R. Spolaor, "The dark side (-channel) of mobile devices: A survey on network traffic analysis," *IEEE Communications Surveys & Tutorials*, Vol. 20, no. 4, Jun 2018, Pages 2658-713.

[18] C. So-In, "A survey of network traffic monitoring and analysis tools," Cse 576m computer system analysis project, Washington University in St. Louis, 2009.

[19] A. Adhikari, S.V. Bianco, L. Denby, L., C.L Mallows, J. Meloche, B. Rao, S.M. Sullivan, Y. Vardi, "Distributed monitoring and analysis system for network traffic," U.S. Patent 7,031,264, 2006.

[20] P. Malloy, A. Cohen, R. Gehl, J. Strohm, R. Elsner, "Interactive network monitoring and analysis," U.S. Patent Application 11/639,863, 2007.

[21] Xiaojun Hei, D. H. K. Tsang and B. Bensaou, "Available bandwidth measurement using Poisson probing on the Internet," IEEE International Conference on Performance, Computing, and Communications, 2004, 2004, Pages 207-214.

[22] J. Liu, C. Zheng, L. Guo, X. Liu, Q. Lu, "Understanding the Network Traffic Constraints for Deep Packet Inspection by Passive Measurement," in Proc. 3rd International Conference on Information Systems Engineering. 2018.

[23] L. de Pedro, M. Martínez Redondo, C. Mancha and J. E. López de Vergara, "Estimating Server Load Based on its Correlation with TCP SYN Response Time", 2020 IFIP Networking Conference (Networking), 2020, Pages 379-385.

[24] D. Salas-González, E.E. Kuruoglu, D.P. Ruiz, "Finite mixture of α-stable distributions," *Digital Signal Processing 19*, Pages 250-264, 2009.

[25] S.A.Broda, M. Hass, J. Krause, M.S. Paolella, S.C. Steude "Stable mixture GARCH models," *Journal of Econometrics 172*, Pages 292-306, 2013.

[26] J. P. Nolan, *Stable Distributions – Models for Heavy Tailed Data*, chap 1. Birkhauser, 2018.

[27] R. N. Rodriguez, "A Guide to Burr Type XII Distributions," Institute of Statistics Mimeo Series No. 1064.

[28] S.A. Abu Bakar, S. Nadarajah, & Z.A.A.K. Adzhar, "Loss modelling using Burr mixtures," *Empir Econ 54*, 1503–1516, 2018.

[29] X.Y. Zhao, M. Fujii, Y. Suganuma, X. Zhao, & Z. Jiang, "Applying the Burr Type XII distribution to decompose remanent magnetization curves," Journal of Geophysical Research: Solid Earth, 123, 8298– 8311, 2018.

[30] Nor Hidayah Binti Ismail and Zarina Binti Mohd Khalid, "EM algorithm in estimating the 2- and 3-parameter Burr Type III distributions," AIP Conference Proceedings 1605, 881, 2014.

[31] M. Tahir, M. Aslam, Z. Hussain, "Bayesian estimation of finite3-component mixture of Burr Type-XII distributions assuming Type-I right censoring scheme," Alexandria Engineering Journal, Volume 55, Issue 4, 2016, Pages 3277-3295, ISSN 1110-0168.

[32] I. Enesi, E. Zanaj, S. Kokonozi and B. Zanaj, "Performance evaluation of stateful load balancing in predicted time intervals and CPU load," IEEE EUROCON 2017 -17th International Conference on Smart Technologies, 2017, Pages 89-94.

[33] Chandra Kopparapu, *Load Balancing Servers, Firewalls, and Caches*, chap 5: Global Server load balancing, Page 73, John Wiley & Sons, Inc. 2002.

[34] H. Nemati, S. V. Azhari and M. R. Dagenais, "Host Hypervisor Trace Mining for Virtual Machine Workload Characterization," 2019 IEEE International Conference on Cloud Engineering (IC2E), 2019, Pages 102-112.

[35] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, A. Brunstrom, "Measuring Latency Variation in the Internet," in Proc. 12th International on Conference on emerging Networking Experiments and Technologies (CoNEXT '16). ACM, New York, NY, USA, Pages 473-480, 2016.

[36] J. Aikat, J. Kaur, F. D. Smith, and K. Jeffay, "Variability in TCP round-trip times," in Proc. 3rd ACM SIGCOMM conference on Internet measurement. ACM, 2003.

[37] D. Perdices, D. Muelas, I. Prieto, L. de Pedro, J. E. López de Vergara, "On the Modeling of Multi-Point RTT Passive Measurements for Network Delay Monitoring," *IEEE Trans. Network and Service Management*, Vol. 16, No. 3, Pages 1157-1169, September 2019.

[38] D. Muelas, J.L. García-Dorado, S. Albandea, J.E. López de Vergara, Javier Aracil, "On the dynamics of valley times and its application to bulk-transfer scheduling," *Computer Communications*, Vol. 164, 2020, Pages 124-137, ISSN 0140-3664.

[39] Avrim Blum, John Hopcroft, & Kannan, Ravi. *Foundations of Data Science*, 2020.

[40] Diego Salas-Gonzalez, Ercan E. Kuruoglu, and Diego P. Ruiz. 2009. "Finite mixture of α-stable distributions," Digit. Signal Process. 19, 2, March 2009, Pages 250–264.

[41] Jeff Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," Technical Report ICSI-TR-97-021, University of Berkeley, 2000.

[42] C. Tomasi, "Estimating Gaussian Mixture Densities with EM – A Tutorial," 2004.

[43] Alan J. Watkins, "An algorithm for maximum likelihood estimation in the three parameter Burr XII distribution," Computational Statistics & Data Analysis, Volume 32, Issue 1, 1999, Pages 19-27.

[44] Muhammad Tahir, Ibrahim M. Almanjahie, Muhammad Abid, Ishfaq Ahmad, "On Estimation of Three-Component Mixture of Distributions via Bayesian and Classical Approaches," Mathematical Problems in Engineering, vol. 2021, Article ID 9944008, 19 pages, 2021.

[45] M. Tahir, M. Aslam, Z. Hussain, "Bayesian estimation of finite3-component mixture of Burr Type-XII distributions assuming Type-I right censoring scheme," Alexandria Engineering Journal, Volume 55, Issue 4, 2016, Pages 3277-3295.

[46] Quanxi Shao, "Notes on maximum likelihood estimation for the three-parameter Burr XII distribution," Computational Statistics & Data Analysis, Volume 45, Issue 3, 2004, Pages 675-687, ISSN 0167-9473.

[47] F.J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American Statistical Association*, Vol. 46, No. 253, Pages 68-78, 1951.

[48] D. Williams, "Weighing the odds: a course in probability and statistics," American Mathematical Monthly 110, 2001, Pages 964-967.

[49] Eduardo Freitas, Assis T. de Oliveira Filho, Pedro R.X. do Carmo, Djamel Sadok, Judith Kelner, "A survey on accelerating technologies for fast network packet processing in Linux environments," Computer Communications, 196, 2002, Pages 148-166.

Luis DE PEDRO SÁNCHEZ is an associate professor at Universidad Autónoma de Madrid (Spain), and president of Naudit HPCN, a company dedicated to high-performance traffic monitoring and analysis. He has been an executive at Hewlett-Packard for more than thirty years. He received his M.Sc. and Ph.D. degrees in Telecommunication Engineering from Universidad Politécnica de Madrid (Spain) in 1987 and 1992, respectively. He currently researches on statistical models for network traffic.

Adrian Mihai ROSU BARBANDEAL is an engineer at Naudit HPCN (Spain). He received his B.Sc. and M.Sc. degrees in Telecommunication Engineering from Universidad Autónoma de Madrid (Spain) in 2019 and 2021, respectively. His research topics are network traffic measurement and network and service monitoring.

Jorge E. LÓPEZ DE VERGARA MÉNDEZ is an associate professor at Universidad Autónoma de Madrid (Spain) since 2007 and is a partner of Naudit HPCN, which is a spin-off company that was founded in 2009 and is devoted to high-performance traffic monitoring and analysis. He received his M.Sc. and Ph.D. degrees in Telecommunication Engineering from Universidad Politécnica de Madrid (Spain) in 1998 and 2003, respectively, where he also held an FPU-MEC research grant. During his Ph.D., he stayed for 6 months in 2000 at HP Labs in Bristol. He studies network and service management and monitoring, and has coauthored more than 100 scientific papers on topics related to this field.