



A queuing theory model for fog computing

Lluís Mas¹ · Jordi Vilaplana¹ · Jordi Mateo¹ · Francesc Solsona¹

Accepted: 18 January 2022 / Published online: 7 February 2022
© The Author(s) 2022

Abstract

Under many scenarios where resources may be scarce or a good Quality of Service is a requirement, appropriately sizing components and devices is one of the main challenges. New scenarios, such as IoT, mobile cloud computing, mobile edge computing or fog computing, have emerged recently. The ability to design, model and simulate those infrastructures is critical to dimension them correctly. Queuing theory models provide a good approach to understanding how a given architecture would behave for a given set of parameters, thus helping to detect possible bottlenecks and performance issues in advance. This work presents a fog-computing modelling framework based on queuing theory. The proposed framework was used to simulate a given scenario allowing the possibility of adjusting the system by means of user-defined parameters. The results show that the proposed model is a good tool for designing optimal fog architectures regarding QoS requirements. It can also be used to fine-tune the designs to detect possible bottlenecks or improve the performance parameters of the overall environment.

Keywords Fog computing · Cloud computing · Simulation · Modelling · Queuing theory

✉ Lluís Mas
lluis.mas@udl.cat

Jordi Vilaplana
jordi.vilaplana@udl.cat

Jordi Mateo
jordi.mateo@udl.cat

Francesc Solsona
francesc.solsona@udl.cat

¹ Department of Computer Science, University of Lleida, Jaume II, 69, Lleida 25001, Spain

1 Introduction

As the Internet of Things (IoT) is rapidly becoming relevant, more and more smart devices can be potentially used to enhance computing power and interactivity in a wide range of scenarios. For example, smart-cities [1], eHealth [2], e-Learning [3], among others.

IoT devices are responsible for 11.5 Zettabytes of all the data generated and this is growing exponentially [4]. These devices have severe computing and storage limitations and they are not able to handle the workload of processing the data they generate. Integrating them to the Cloud fixes that issue but it is not optimal. The recent adoption of Fog and Edge paradigms dramatically improves the whole system's performance by reducing the amount of data that needs to be transmitted to and processed by the Cloud. However, the application requirements should be considered before designing the architecture to achieve the best performance. The system architecture combines IoT sensors, fog devices and cloud infrastructures. All of these need to be considered to obtain the best availability and performance [5].

Mobile cloud computing (MCC) supplies new services and facilities to enable mobile users (MUs) to take full advantage of cloud computing [6]. However, the cloud is usually located far from the MUs, which results in a high network latency. This inevitably reduces the quality of user service (QoS). To solve the network latency issue, a new paradigm called mobile edge computing (MEC) has been proposed [7]. MEC has become a key technology for the Internet of Things and 5G [8]. It can be regarded as a special example of MCC. A cloudlet is a type of edge server that provides various services to users in close proximity to mobile devices (MDs) [9]. That means it can reduce the latency and energy consumption by offloading workflow applications (WAs) to a set of cloudlets.

Mobile Edge Computing or Fog computing refers to offloading computationally intensive algorithms from a mobile device to the cloud or an intermediate cloud in order to save resources, e.g., time and battery in the mobile device [10]. Thus, fog computing and fog architectures can improve the Quality of Service (QoS) of a system, in other words, low latency and delays, offering several performance improvements when dealing with applications that rely on several devices [11]. By streaming data from the mobile devices to the fog layer in real time to pre-process the data, latency and network usage can be reduced, thus providing a higher QoS.

Research in the fog computing field is just starting and it is expected to become a unifying platform, rich enough to deliver this new breed of emerging services and enable the development of new applications [12]. Many existing platforms incorporate new devices to enhance the communication with users and automate the data collection process [13]. As these devices become more relevant and abundant, existing cloud models and platforms need to be redesigned, taking into account the fog and IoT paradigms. Fog architectures can induce a great improvement for institutions, organisations and companies, facilitating real-time monitoring, critical decision-making, and optimising resources. Overall, scalability and fares are improved.

Data collected from mobile devices can be made accessible to overall users in real time by the use of fog computing, where a requirement is to design systems

that can offer a high QoS given an increasing number of devices connected to the system is a requirement. These systems need to have high scalability, ensuring that they can keep up with an increasing demand when more devices are added, while maintaining the level of QoS established in the service-level agreement (SLA). The SLA is the contract that specifies the QoS that a provider has to offer for the price being paid [14].

Most of the publications related to fog computing deal with load prediction, simulation, scheduling, load balancing, etc. [15, 16]. However, there is a lack of work focused on designing cloud-fog architectures properly. In this context, the key contribution of this work is to provide a generic model for designing efficient fog architectures, depending on the expected workload.

To address fog and cloud design challenges and to guarantee the QoS requirements at a minimal resource cost, this paper makes the following contributions:

- Modeling cloud-fog architectures using a queuing theory model.
- Propose a model capable of evaluating the performance regarding response time, throughput and node usages metrics aimed to guarantee QoS at minimal resource cost.

Therefore, this work aims to develop a user-friendly generic model, based on queuing theory, to help design two-level cloud-fog frameworks. This two-level proposal (cloud and fog) is an extension of the model for a single cloud level, presented in [17].

Moreover, the model proposed is aimed to assure the QoS performance agreed in an SLA by using configurable parameters. Furthermore, the model proposed can be applied to evaluate scenarios and fine-tune the parameters adapting the architecture to the designer's needs. Finally, the model presented is flexible and is not limited to any specific kind of cloud.

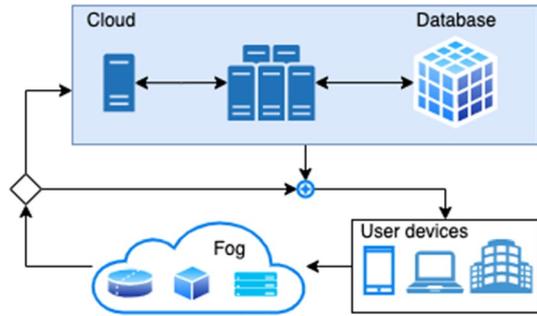
To sum up, this work presents a number of significant contributions in the design and performance of cloud-fog architectures. These improvements in performance have a direct impact on reducing infrastructure costs and ensuring a greener footprint.

Section 2 introduces previous work. The proposed fog model is explained in Sect. 3. Section 4 presents the model implemented and examples showing its applicability on tuning the fog in order to fit the agreed QoS performance. Section 5 presents the outcomes obtained from the work. The discussion and comparison with previous work is presented in Sect. 6. Finally, in Sect. 7, the results presented are summarised together with the conclusions obtained, and possible future research lines for this work.

2 Related work

Queuing theory has been used to analyse the performance of cloud services [18]. Previous work has been done on modelling and simulating fog models with the Java Modeling Tool (JMT) for Internet of Things middle-ware [19]. There are

Fig. 1 Fog architecture. For a Mobile Cloud Computing environment (MCC), Fog should be replaced by Cloudlets



also tools to perform analysis for cloud and fog computing [20]. Queuing theory has also been used to enhance job completion by means of load balancing [10], and to reduce data overhead for heterogeneous networks in smart city environments [21].

Mobile device users usually connect to cloudlets through WiFi. These data centres are widely distributed, which means they are in different geographic points. Because of that, the decision about which cloudlet to connect to becomes a problem as it not only depends on proximity, but also load balancing [22]. Adding an extra layer (cloud and fog) makes it more complex, as suggested in [23]. In [24], queuing theory and multi-objective optimisation methods were employed to determine the offloading solutions for the deadline-constrained workflows in the cloudlet-based mobile cloud, a key challenge in these environments since a cloudlet has limited resources.

Various algorithms for optimal assignment of tasks in Mobile Cloud Computing environment (MCC) have been proposed [22, 25]. The main feature of these algorithms is to efficiently assign tasks depending on the location of the cloud environment. Another main concern is the optimisation of the energy consumption for all mobile devices [26]. That is the terminology used in MCC for designing cloud/fog architectures as well as the one shown in Fig. 1. The cloud (public) is situated far from the users. Therefore, the communication between the cloud servers and the users requires an Internet connection [27], consequently, having a slow connection would make the task spend considerable amounts of time and battery. Queuing theory, control theory, and machine learning are the main solutions for this issue [28, 29], equivalent to having only a one-level architecture (just the cloud). Queuing models have proven useful for developing analytical models to provide QoS in cloud architectures [17, 30–32].

There are many examples of practical proposals for one-level architecture. For instance, in [8], the authors were interested in designing a virtualised mobility management entity (vMME) for 5G Mobile Networks, but on a single level (the cloud). The proposal determines the minimum number of processing instances required to guarantee a given mean system response time. To accomplish that, the authors used an approach based on queuing theory. Another important application is in web auto-scaling [33], where cloud service providers and researchers are working to reduce the costs while maintaining the Quality of Service (QoS).

Also, in [34], the authors allocate Virtual Machines dynamically to obtain an optimal VM utilisation and perform traffic control at the same time.

There are also many practical applications of fog computing. In [35], the authors proposed an architecture, based on the queuing-theory model presented in [17], for managing Unmanned Aerial Vehicles (UAV). In [36], the authors performed thorough research on fog applications; they pointed out their uses in many fields including health care system [37, 38], augmented reality [39], traffic control systems [40], and video streaming systems [41]. The examples of fog computing are not limited to these. On the contrary, its applicability keeps growing day by day.

3 Fog model

Many computational systems and data centers have multiple components. A system where interactions are performed by the users is assumed in this model. These communications are accomplished by exchanging data with several devices (smartphones, smart bands, smartwatches, health sensors, laptops, personal computers, etc.) with the cloud platform. In order to improve cloud QoS, such dynamic information should be managed by an intermediate stage of the computing environment, called Fog.

Figure 1 shows a diagram of a fog computing architecture. The components of such architecture are the computing servers and the database on the cloud. The smart devices used to exchange and provide data are in the middle, forming the fog. Finally, the users engaged with the system are represented on the bottom level.

The fog architecture shown in Fig. 1 is modelled by queuing theory. The model obtained, derived from the work published in [17], is shown schematically in Fig. 2. It follows a closed Jackson network [42], with a fixed number of jobs circulating inside, starting from the user devices. The system workflow is governed by a probabilistic routing through the network. Since there are no outside arrivals nor departures from the network, the equilibrium distribution can be mathematically formalised. This model represents a system with a fixed number of users, and the smart devices of the fog are designed to enhance the performance of computation and data exchange between the user devices and the cloud. The purpose of this architecture is to analyse the behaviour of the system for a fixed workflow. The fog model is made up of three components, the client devices, the fog and the cloud. The clients ($CS_i, i = 1..M$) send the workload to the fog. The fog nodes ($FS_j, j = 1..N$) are responsible for processing the client's tasks or resending them to the cloud in case it has not enough computing power to process the data. The fog is also responsible for collecting data and processing it asynchronously. In the proposed model, the fog will forward the workflow to the cloud with a probability of κ .

Based on the proposal presented in [17], the cloud level consists of 4 interconnected components, the entry server (ES), processing servers (PS), database server (DS) and, the output server (OS). The ES is the entry point to the cloud. From there, the load-balancer is in charge of distributing the data towards the processing servers. Each processing server (PS_k) accesses the DS with a probability of δ . The DS is responsible for servicing the data (data warehouse, regular

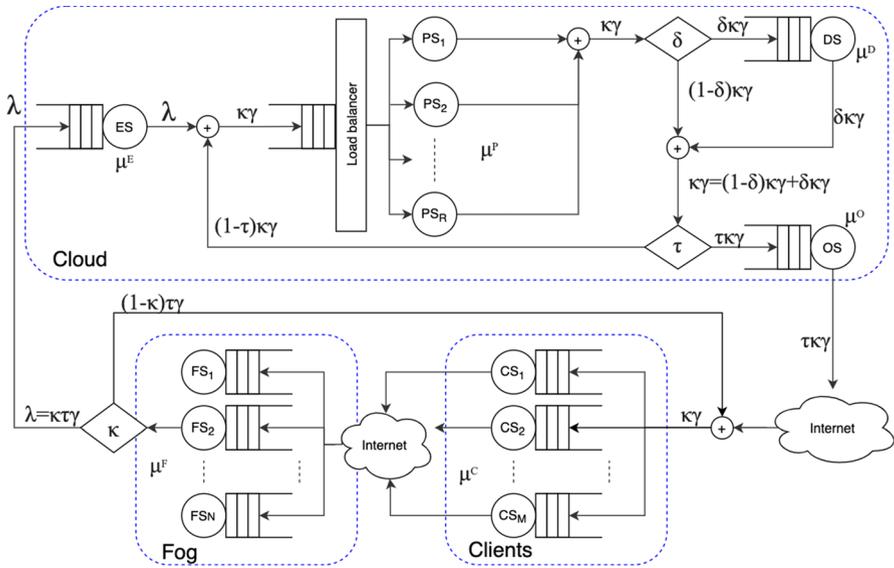


Fig. 2 Fog model using queuing theory

files, operational database, etc.). Finally, the OS is left in charge of transmitting the processed data and results to the Internet, acting as an output gateway of the cloud subsystem. The workflow from the Internet enters the corresponding client, thus ending the closed Jackson network.

The ES, DS and OS are modelled as *M/M/1* queues, relying on three exponential probability density functions with service rates μ^E , μ^D and μ^O , respectively. Every CS_i is modelled as an *M/M/1* queue with the same exponential service rates, that is, $\mu_i^C = \mu^C, i = 1..M$. The fog servers are modelled as an *M/M/N* queue with $\mu_j^F = \mu^F, j = 1..N$. In other words, all the fog servers have the same computational power. In the cloud, the processing serves are also modelled as an *M/M/R* queue, where R is the number of processing servers. All the processing servers are also considered to have the same processing capabilities, that is ($\mu_i^P = \mu^P, i = 1..R$).

Jobs cannot circulate freely through the network. Three routers divide the workflow. One of them is located after the Fog, where the workflow can be sent to the cloud or backwards to the clients. Jobs will be redirected to the cloud with probability κ , representing high cloud computing or data requirements. The Fog will have enough processing or data management capabilities to serve the client jobs with a probability $1 - \kappa$. Another workflow routing is determined by δ , representing the probability of accessing the cloud database. Finally, jobs will exit the cloud with a probability τ . In such a case, the jobs will be redirected to the output server (OS), responsible for forwarding jobs to the clients through the Internet. Conversely, jobs will be redirected to the processing servers with a probability of $1 - \tau$, representing jobs that request more computing or data before they are finished and ready to be passed to the clients.

Fig. 3 Transition probabilities matrix

$$\mathbf{T} = \begin{array}{c} \begin{array}{l} ES \\ PS \\ DS \\ OS \\ CS \\ FS \end{array} \begin{array}{l} ES \\ PS \\ DS \\ OS \\ CS \\ FS \end{array} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & (1-\delta)(1-\tau) & \delta & (1-\delta)\tau & 0 & 0 \\ 0 & (1-\tau) & 0 & \tau & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \kappa & 0 & 0 & 0 & (1-\kappa) & 0 \end{pmatrix} \end{array}$$

The workflow is defined by the stochastic transition matrix (T) (Fig. 3). This matrix defines the one-step transition probability of jobs between a pair of adjacent servers of the architecture proposed. This way, jobs always move forward from ES to PS , from OS to CS and from CS to FS . Jobs also move forward from PS to DS with probability δ , to OS with probability τ , and backwards to PS with probability $(1 - \tau)$. Note that the probabilities of each row must sum 1.

4 Simulation

The following section presents the simulation of the outcomes obtained to evaluate our proposal (see Fig. 4). It implements a fog architecture modelled by queuing theory as presented above (Section 3). The model was implemented using the statistical R [43] software (version 4.0.1) and the *queueing* package [44]. The model was deployed as an R Shiny application [45] and is accessible using any modern web browser. The developed software enables an analysis of how the performance is affected when different parameters are modified. That way, a designer can assess whether the design they came up with will handle the workload or, perhaps, the architecture allocates too many resources.

The software enables an analysis of how the response time is affected by modifying the different metrics of the model to be performed, allowing verification of whether the model behaves as expected when a range of parameters and system configurations is tested in the simulation environment. The source code of the simulation is publicly available at GitHub.¹

4.1 Input parameters

The model's input parameters are:

- J : Number of jobs (customers) in the system.
- μ^E : Service rate of the Entry Server.
- μ^P : Service rate of every Processing Server.
- μ^D : Service rate of the Database Server.
- μ^O : Service rate of the Output Server.

¹ Source code hosted at GitHub: <https://github.com/jvilaplana/QTFM/releases/tag/v1.0.0>.

Queuing Theory Model for Fog Computing

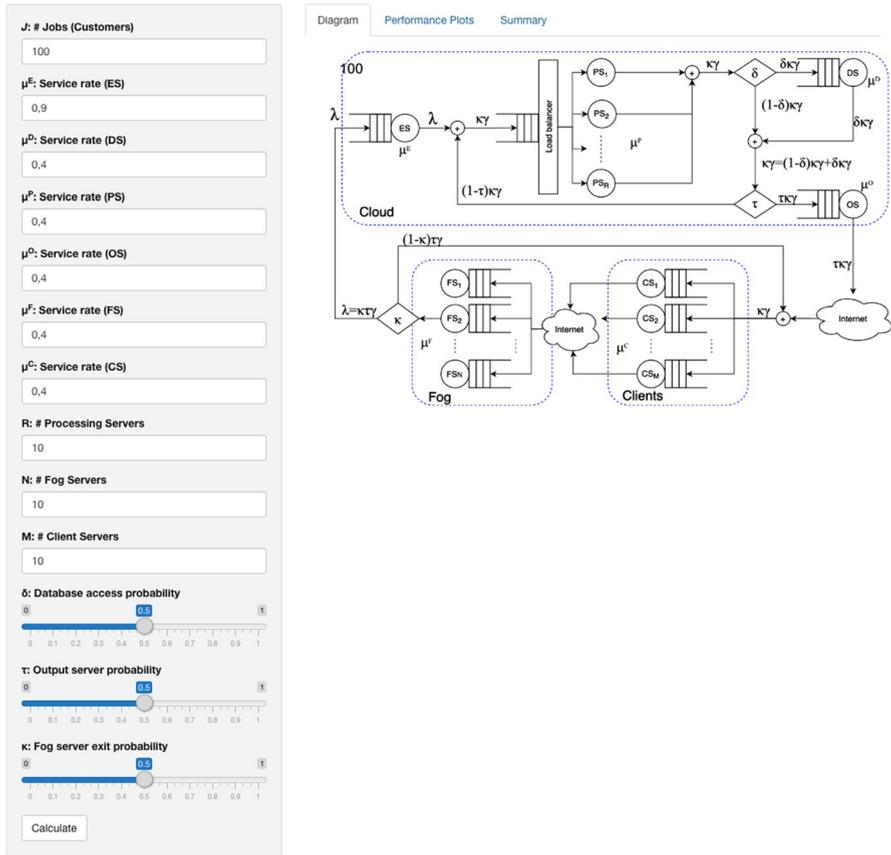


Fig. 4 Overview of the model’s website

- μ^F : Service rate of every fog Server.
- μ^C : Service rate of every Client Server.
- R : Number of processing servers.
- N : Number of fog servers.
- M : Number of client servers.
- δ (0...1): Database access probability.
- τ (0...1): Output server exit probability.
- κ (0...1): Fog server exit probability.

Once the parameters are set, the simulator automatically simulates the closed Jackson network model specified in the transition matrix \mathbf{T} until an equilibrium of the overall system is reached.

4.2 Outcomes

When the simulation ends, the obtained QoS outcomes are displayed, these being throughput, response time, and node usage. Throughput is defined as the number of jobs processed per unit of time. Response time is the mean elapsed time per job. Finally, node usage is the percentage of occupation of each server.

Furthermore, a summary table containing additional information is displayed. Said table has a row for every node and has the following columns:

- **L**: Mean number of customers of the network.
- **W**: Mean time spent in the network.
- **X**: Total throughput of the network.
- **Lk**: Mean number of customers in each node (server).
- **Wk**: Mean time spent in each node.
- **Xk**: Throughput of each node.
- **ROk**: Usage of each node.

All of these parameters can be used to verify the overall degree of occupation of the components of the architecture. These outcomes allow the deployment of additional resources in the appropriate subsystems to be properly optimised (either Cloud, Client, or Fog).

5 Results

In this section, we present the results guided by an example based on a real-life scenario. We show how, for this example, the resource needs are minimised while the QoS requirements are still guaranteed.

5.1 Case study

The case study consists of a system with 100 jobs. Every job has a 50% probability of needing database access, 50% chance that the task is computing-heavy, hence needing more cloud computing time and 50% chance of being a fog heavy-task, needing more fog computing time.

This scenario dictates that a mean response time of up to 4 s is acceptable. Anything past that is considered to violate the quality of service agreement.

After a first design, an architect assessed that the system needs 10 Processing servers of 0.4 service rate, 10 Fog servers of 0.4 service rate, 10 Client servers of 0.4 service rate. The service rate of the Entry server should be 0.9 the Database server should have a 0.4 service rate finally, the service rate of the output server should be 0.4.

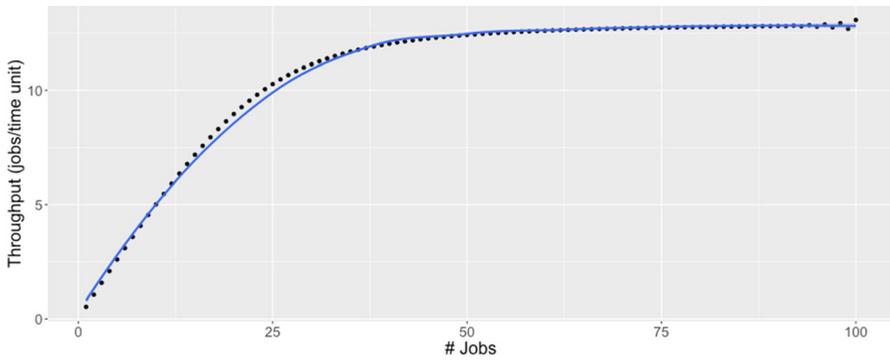


Fig. 5 Throughput evolution plot. Relationship between the number of jobs and throughput metric in the case study

Now let us assess how this initial design will fare in the real scenario using the proposed model. As well as whether some optimisations could be made to the design.

In the following subsections, different metrics are analysed to evaluate the performance of this case study regarding throughput, response time and node usage. Finally, by comparing the results presented in this case study between using the proposed model and one of the most popular simulators in the field (iFogSim), we hope to determine the strengths and advantages of using our model and highlight the situations in which our model could obtain better results and its usage is recommended.

5.2 Throughput

Throughput is a metric commonly used to measure performance. It is defined as the ratio of the number of the number of jobs served per unit of time. This is, $R = J/T$, where R is the throughput, J is the number of jobs, and T is the time unit. Figure 5 shows a plot of the throughput from 0..100 jobs, as specified in the parameter settings (see Fig. 4).

This plot can be used to assess the limit of the system and its capacity for processing jobs. Note that the x -axis represents the number of jobs in the system, whereas the y -axis represents the value of the throughput metric. This plot can be used to assess the limit of the system and its ability for processing jobs. That informs the system-architect designer of the processing capacity of the entire system. Thus, evaluating this metric allows cloud-architect to minimize breaches of Service-Level Agreement (SLA) contracts. SLA is usually related to service completion limits.

5.3 Response time

Response time is another standard metric to assess the performance of a system. It is defined as the elapsed time between submitting a request to the server and completing the task execution. The total system response time ($T_{response}$)

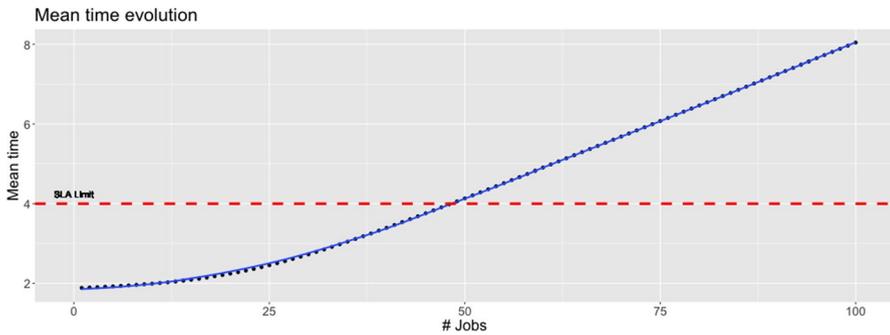


Fig. 6 Response time evolution plot. Relationship between the number of jobs, the SLA guarantees and the response time metric in the case study

corresponds to the summatory of the residence times at the various nodes of the model, this is $T_{response} = \sum_{k=1}^K R_k$, where K is the total number of nodes and R_k is the total response time of a job in the node k .

Properly calculating the response time is of utmost importance for complying with any SLA contract, and to ensure acceptable performance of the final designed fog architecture. Figure 6 shows the response time of the entire system when varying the number of jobs from 0...100, as shown above in the throughput case. Similarly, the x -axis represents the number of jobs in the system, whereas the y -axis represents the value of the response time metric.

As can be appreciated, the response time increases almost linearly. We assume the system is in ideal conditions, in other words, the additional overhead caused by saturating the nodes and connections of the system is negligible. Processing saturation will be reached when overloading the servers. Connection saturation would be reached instead when the tasks exceed the bandwidth capacity. Simulating saturation caused by other factors is beyond the scope of this work as it would reduce the scalability of the model. The model is focused on computing resource usage and optimising these accordingly. In addition, we are not interested in measuring or designing particular aspects of connections and servers. Thus, no worsening of the response time with the number of tasks can be observed. That means that the response time increases linearly with the number of tasks and no polynomial or exponential extra cost penalties are introduced by increasing them.

In addition to the evolution of the response time as the number of jobs increases to the specified target, Fig. 6 also includes a dashed line with the specified SLA, showing the maximum response time that would be acceptable for the system. The intersection between the SLA and response timelines determines the maximum number of jobs that the modelled system would be able to handle without affecting the desired quality of service.

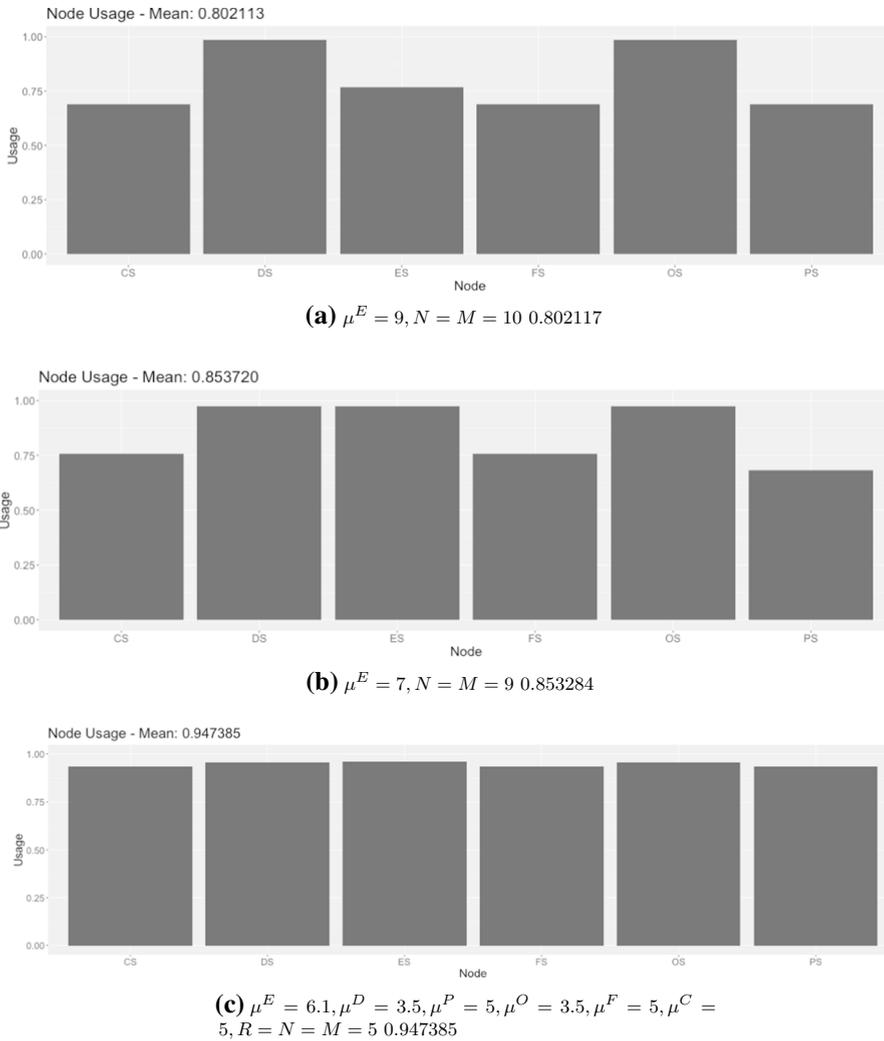


Fig. 7 Sensitivity analysis of nodes usage executing the workload while tuning the main parameters

5.4 Node usage

This subsection describes the results related to the percentage of resources consumed by the case study workload, which shows how active the system’s nodes are. Figure 7 depicts the occupation of every node for the target number of jobs and the given parameters.

Note that the x -axis represents the different nodes [entry server (ES), processing servers (PS), database server (DS) and the output server (OS)]. In contrast, the y -axis represents the percentage of usage normalized between 0 and 1. The values close to 0 illustrate that the node is underused, whereas around 1 mean that the node

	L	W	X	Lk	Wk	Xk	ROk
Net	100	27.19762	3.676792	NA	NA	NA	NA
Nd1	NA	NA	NA	0.9883262	2.209057	0.4473973	0.49710811
Nd2	NA	NA	NA	1.8709405	2.500008	0.7483737	0.18709342
Nd3	NA	NA	NA	32.1545582	82.351334	0.3904558	0.97613956
Nd4	NA	NA	NA	60.7155404	152.325677	0.3985903	0.99647580
Nd5	NA	NA	NA	2.1149691	2.500000	0.8459876	0.02114969
Nd6	NA	NA	NA	2.1556657	2.500028	0.8622566	0.21556415

Fig. 8 Usage of each node

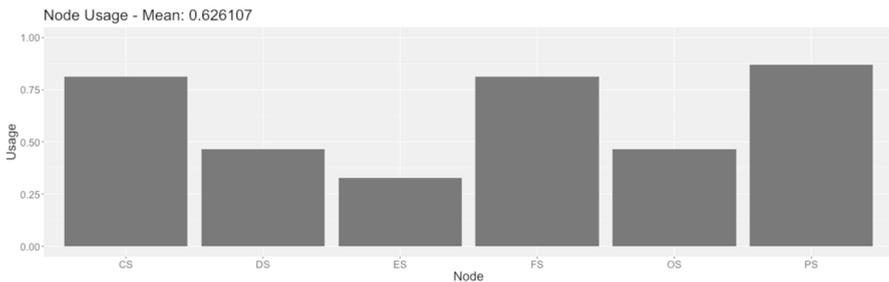


Fig. 9 Percentage of resources consumed in each node by running DCNSFog example

is saturated. Thus, this plot allows us to quickly check which components of the architecture are being underused and which others are saturated. These plots are remarkably useful for detecting bottlenecks in the system, in other words, which nodes are saturated and also which are redundant or oversized.

Figure 7a shows how some of the nodes are slightly underused. Fine-tuning some of the model parameters, like the service rate of the entry server ($\mu^E : 0.9 \rightarrow 0.7$), and the number of fog and client servers ($N, M : 10 \rightarrow 9$) resulted in more optimal utilization of these nodes, as shown in Fig. 7b. With a thorough parameter adjustment, it can be seen how node usage changes from 85% (Fig. 7b) to a 94% (Fig. 7c).

Figure 8 shows the additional information in a Summary table produced for the example in Fig. 4. These values are additional aid for tuning the architecture as a debugging aid.

5.5 Model validation

The developed Fog Model has been tested against iFogSim [46]. iFogSim is a state-of-the-art simulator widely used in the community. It uses traces to simulate the architecture. The main objectives of the comparison are to check how the system fares in iFogSim then, run the architecture in our model, detect bottlenecks or systems with too many resources allocated. Once these issues are detected, the system should be modified to improve node usage.

To achieve that, iFogSim has been modified to include node usage at the end of each simulation.

The base for this study has been the DCNSFog example that base iFogSim provides. The example uses four areas, each one containing one fog device (FS) and four camera devices (CS).

Running the example gives an execution time of 728 s and a total execution cost of 25,759. Figure 9 shows the analysis of the same problem using the proposed Fog Model. As it can be seen in Fig. 9, the entry server (ES) and the output server (OS) allocate too many resources.

After reducing the resources allocated by the Entry and Output servers in iFogSim, we obtained an execution time of 737 s and a cost of 19,004. In summary, it can be appreciated how the results yielded by both of the platforms correlate. Finally, using the proposed Fog Model, resource usage can be vastly improved.

6 Discussion

Starting from an initial work where a one-level cloud environment was modelled by using queuing theory [17], the model was expanded to include a two-level fog system.

The main advantage of our proposal, as opposed to other fog simulation software like CloudSim [47], EdgeCloudSim [48] and iFogSim [46], is that the parameters of the model can be adjusted in real-time, allowing users to fine-tune the architecture design in advance and detect how it would behave when changing the system parameters. This model can be used for rapid prototyping of fog architectures according to desired performance metrics (QoS or/and SLA). Moreover, the parameters in the model presented, like the number of servers and their computing capabilities, can be adjusted in real-time allowing the optimal amount of resources to be obtained. Consequently, all the nodes will be close to total computing occupation without being saturated. While developing the presented work, user-friendliness and ease of use of the application was a top priority as well. The final should be able to adjust the parameters of the model in real-time without the need for additional programming.

In the example presented, an increase in the utilisation of 14% was obtained by changing parameters only manually, either using lower performing machines or decreasing the number of these. Despite reducing the amount of computing power the system is provided with, the SLA requirements were still met, thus reducing the price a provider would pay for its fog infrastructure while simultaneously ensuring a greener footprint.

Rather than a simulation software, as the related literature presents, our proposal consists of a framework that will ease the design of fog architectures by simulating their load. The designer will easily spot bottlenecks, hence allowing the fine-tuning of the system by reducing or increasing the number of servers or their computing power. On top of that, the overcomes can be seen as the best configuration at a given time with a given load. Prediction and optimisation algorithms can be applied on top of the work presented to tune the architecture dynamically depending on the predicted load.

7 Conclusions

Modelling and simulating fog computing systems is a complex task. Queuing theory models can help perform a first approach to those simulations, helping to dimension their components properly. Systems adopting the Internet of Things paradigm can be successfully modelled as fog architectures. In this paper, we proposed software capable of modelling several fog computing architectures based on a closed Jackson network employing queuing theory, which will aid architects to design efficient fog architectures.

The results showed that useful performance metrics can be obtained to gather insight into how to dimension the system properly before designing and deploying an actual fog architecture with QoS or SLA constraints.

Future work includes performing more testing and validating the model used with other work from the literature. Also, an optimization algorithm could be developed to find the best parameters to ensure the optimal node usage automatically, instead of having to fine-tune the parameters manually. Additionally, a load prediction algorithm could be run on top of it so that the load parameters (job number and access probabilities) could also feed the optimisation algorithm not only to predict the load a system will have but also what architecture will best fit it without saturating while still guaranteeing the QoS constraints. Moreover, more work needs to be done to extend the simulation software to implement more QoS metrics.

Acknowledgements This work was supported by the Ministerio de Economía y Competitividad under contract TIN2017-84553-C2-2-R and the Ministerio de Ciencia e Innovación under contract PID2020-113614RB-C22. Dr. Jordi Vilaplana is a Serra Húnter Fellow.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Tang B, Chen Z, Hefferman G, Wei T, He H, Yang Q (2015) A hierarchical distributed fog computing architecture for big data analysis in smart cities. In: ACM International Conference Proceeding Series, vol 07-09-October. <https://doi.org/10.1145/2818869.2818898>
2. Qi J, Yang P, Min G, Amft O, Dong F, Xu L (2017). Advanced internet of things for personalised healthcare systems: a survey. <https://doi.org/10.1016/j.pmcj.2017.06.018>
3. Pecori R (2018) A virtual learning architecture enhanced by fog computing and big data streams. Future Internet. <https://doi.org/10.3390/fi10010004>
4. Forum WE (2019) How much data is generated each day? World Economic Forum. <https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/>

5. Santos GL, Takako Endo P, da Silva Ferreira, Lisboa Tigre MF, Ferreira da Silva LG, Sadok D, Kelner J, Lynn T (2018) Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures. *J Cloud Comput.* <https://doi.org/10.1186/s13677-018-0118-3>
6. Peng K, Zhu M, Zhang Y, Liu L, Zhang J, Leung VCM, Zheng L (2019) An energy- and cost-aware computation offloading method for workflow applications in mobile edge computing. *Eurasip J Wirel Commun Netw.* <https://doi.org/10.1186/s13638-019-1526-x>
7. Mach P, Becvar Z (2017) Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tutor* 19(3):1628–1656. <https://doi.org/10.1109/COMST.2017.2682318>
8. Prados-Garzon J, Ramos-Munoz JJ, Ameigeiras P, Andres-Maldonado P, Lopez-Soler JM (2017) Modeling and dimensioning of a virtualized MME for 5G mobile networks. <https://doi.org/10.1109/TVT.2016.2608942>
9. Satyanarayanan M, Bahl P, Cáceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Comput* 8(4):14–23. <https://doi.org/10.1109/MPRV.2009.82>
10. Sthapit S, Thompson J, Robertson NM, Hoppood JR (2019) Computational load balancing on the edge in absence of cloud and Fog. *IEEE Trans Mob Comput* 18(7):1499–1512. <https://doi.org/10.1109/TMC.2018.2863301>
11. Ramalho F, Neto A, Santos K, Filho JB, Agoulmine N (2015) Enhancing eHealth smart applications: a Fog-enabled approach. In: 2015 17th International Conference on E-Health Networking, Application and Services, HealthCom 2015, pp 323–328. <https://doi.org/10.1109/HealthCom.2015.7454519>
12. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: MCC'12—Proceedings of the 1st ACM Mobile Cloud Computing Workshop, pp 13–15. <https://doi.org/10.1145/2342509.2342513>
13. Mateo-Fornés J, Pagès-Bernaus A, Plà-Aragónés LM, Castells-Gasia JP, Babot-Gaspa D (2021) An internet of things platform based on microservices and cloud paradigms for livestock. *Sensors.* <https://doi.org/10.3390/s21175949>
14. Vilaplana J, Mateo J, Teixidó I, Solsona F, Giné F, Roig C (2015) An SLA and power-saving scheduling consolidation strategy for shared and heterogeneous clouds. *J Supercomput* 71(5):1817–1832. <https://doi.org/10.1007/s11227-014-1351-2>
15. Farooq U, Shabir MW, Javed MA, Imran M (2021) Intelligent energy prediction techniques for fog computing networks. *Appl Soft Comput* 111:107682. <https://doi.org/10.1016/j.asoc.2021.107682>
16. Dixit A, Yadav AK, Kumar S (2017) An efficient architecture and algorithm for server provisioning in Cloud computing using clustering approach. In: Proceedings of the 5th International Conference on System Modeling and Advancement in Research Trends, SMART 2016, vol 8(1), pp 260–266. <https://doi.org/10.1109/SYSMART.2016.7894532>
17. Vilaplana J, Solsona F, Teixidó I, Mateo J, Abella F, Rius J (2014) A queuing theory model for cloud computing. *J Supercomput* 69(1):492–507. <https://doi.org/10.1007/s11227-014-1177-y>
18. Liu X, Li S, Tong W (2015) A queuing model considering resources sharing for cloud service performance. *J Supercomput* 71(11):4042–4055. <https://doi.org/10.1007/s11227-015-1503-z>
19. Rathod D, Chowdhary DG (2019) Scalability of M/M/c queue based cloud-fog distributed internet of things middleware. *Int J Adv Netw Appl* 11(01):4162–4170. <https://doi.org/10.35444/ijana.2019.11015>
20. Tadakamalla U, Menasce D (2019) FogQN: an analytic model for fog/cloud computing. In: Proceedings—11th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018, pp 307–313. <https://doi.org/10.1109/UCC-Companion.2018.00073>
21. Said O, Tolba A (2020) DORS: a data overhead reduction scheme for hybrid networks in smart cities. *Int J Commun Syst.* <https://doi.org/10.1002/dac.4435>
22. Sundararaj V (2019) Optimal task assignment in mobile cloud computing by queue based Ant-Bee algorithm. *Wirel Pers Commun* 104(1):173–197. <https://doi.org/10.1007/s11227-018-6014-9>
23. Maiyama KM, Kouvatso D, Mohammed B, Kiran M, Kamala MA (2017) Performance modeling and analysis of an OpenStack IaaS cloud computing platform. In: Proceedings—2017 IEEE 5th International Conference on Future Internet of Things and Cloud, FiCloud 2017, vol 2017-Janua, pp 198–205. <https://doi.org/10.1109/FiCloud.2017.54>
24. Xu X, Fu S, Yuan Y, Luo Y, Qi L, Lin W, Dou W (2019) Multiobjective computation offloading for workflow management in cloudlet-based mobile cloud using NSGA-II. *Comput Intell* 35(3):476–495. <https://doi.org/10.1111/coin.12197>

25. Rashidi S, Sharifian S (2017) A hybrid heuristic queue based algorithm for task assignment in mobile cloud. *Future Gener Comput Syst* 68:331–345. <https://doi.org/10.1016/j.future.2016.10.014>
26. Pandi V, Perumal P, Balusamy B, Karuppiah M (2019) A novel performance enhancing task scheduling algorithm for cloud-based e-health environment. *Int J E-Health Med Commun* 10(2):102–117. <https://doi.org/10.4018/IJEHMC.2019040106>
27. Sundararaj V, Muthukumar S, Kumar RS (2018) An optimal cluster formation based energy efficient dynamic scheduling hybrid MAC protocol for heavy traffic load in wireless sensor networks. *Comput Secur* 77:277–288. <https://doi.org/10.1016/j.cose.2018.04.009>
28. Zhang J, Huang H, Wang X (2016) Resource provision algorithms in cloud computing: a survey. *J Netw Comput Appl* 64:23–42. <https://doi.org/10.1016/j.jnca.2015.12.018>
29. El Kafhali S, Salah K (2018) Modeling and analysis of performance and energy consumption in cloud data centers. *Arab J Sci Eng* 43(12):7789–7802. <https://doi.org/10.1007/s13369-018-3196-0>
30. Palvannan RK, Teow KL (2012) Queueing for healthcare. *J Med Syst* 36(2):541–547. <https://doi.org/10.1007/s10916-010-9499-7>
31. Bai WH, Xi JQ, Zhu JX, Huang SW (2015) Performance analysis of heterogeneous data centers in cloud computing using a complex queueing model. *Math Probl Eng*. <https://doi.org/10.1155/2015/980945>
32. Cassar MR, Borg D, Camilleri L, Schembri A, Anastasi EA, Buhagiar K, Callus C, Grech M (2021) A novel use of telemedicine during the COVID-19 pandemic. *Int J Infect Dis* 103:182–187. <https://doi.org/10.1016/j.ijid.2020.11.170>
33. Singh P, Gupta P, Jyoti K, Nayyar A (2019) Research on auto-scaling of web applications in cloud: survey, trends and future directions. *Scalable Comput* 20(2):399–432. <https://doi.org/10.12694/scpe.v20i2.1537>
34. Hanini M, El Kafhali S, Salah K (2019) Dynamic VM allocation and traffic control to manage QoS and energy consumption in cloud computing environment. *Int J Comput Appl Technol* 60(4):307–316. <https://doi.org/10.1504/IJCAT.2019.101168>
35. Luo F, Jiang C, Yu S, Wang J, Li Y, Ren Y (2017) Stability of cloud-based UAV systems supporting big data acquisition and processing. *IEEE Trans Cloud Comput* 7(3):866–877. <https://doi.org/10.1109/tcc.2017.2696529>
36. Rahman G, Chuah CW (2018) Fog computing, applications, security and challenges, review. *Int J Eng Technol* 7:1615
37. Brzozza-Woch R, Konieczny M, Kwolek B, Nawrocki P, Szydło T, Zieliński K (2015) Holistic approach to urgent computing for flood decision support. *Procedia Comput Sci* 51(1):2387–2396. <https://doi.org/10.1016/j.procs.2015.05.414>
38. Cao Y, Chen S, Hou P, Brown D (2015) FAST: a fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In: *Proceedings of the 2015 IEEE International Conference on Networking, Architecture and Storage, NAS 2015*, pp 2–11. <https://doi.org/10.1109/NAS.2015.7255196>
39. Dastjerdi AV, Buyya R (2016) Fog computing: helping the internet of things realize its potential. *Computer* 49(8):112–116. <https://doi.org/10.1109/MC.2016.245>
40. Tang C, Xia S, Zhu C, Wei X (2019) Phase timing optimization for smart traffic control based on fog computing. *IEEE Access* 7:84217–84228. <https://doi.org/10.1109/ACCESS.2019.2925134>
41. Chen N, Chen Y, You Y, Ling H, Liang P, Zimmermann R (2016) Dynamic urban surveillance video stream processing using fog computing. In: *Proceedings—2016 IEEE 2nd International Conference on Multimedia Big Data, BigMM 2016*, pp 105–112. <https://doi.org/10.1109/BigMM.2016.53>
42. Khac CN, Thanh KB, Dac HH, Hong SN, Tran VP, Cong HT (2019) An open Jackson network model for heterogeneous infrastructure as a service on cloud computing. *Int J Comput Netw Commun* 11(1):63–80. <https://doi.org/10.5121/ijcnc.2019.11104>
43. Core Development Team R (2020) A Language and Environment for Statistical Computing. <http://www.r-project.org>
44. Jiménez PC, Montoya YR (2017) queueing: A package for analysis of queueing networks and models in R. *R Journal* 9(2):116–126. <https://doi.org/10.32614/rj-2017-051>
45. Vinet L, Zhedanov A (2011) A ‘missing’ family of classical orthogonal polynomials, vol 44. Packt Publishing. <https://doi.org/10.1088/1751-8113/44/8/085201>
46. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R (2017) iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Softw Pract Exp* 47(9):1275–1296. <https://doi.org/10.1002/spe.2509>

47. Ahmad MO, Khan RZ (2019) Cloud computing modeling and simulation using CloudSim environment. *Int J Recent Technol Eng* 8(2):5439–5445. <https://doi.org/10.35940/ijrte.B3669.078219>
48. Sonmez C, Ozgovde A, Ersoy C (2018) EdgeCloudSim: an environment for performance evaluation of edge computing systems. *Trans Emerg Telecommun Technol*. <https://doi.org/10.1002/ett.3493>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.