Randomized Load Balancing under Loosely Correlated State Information in Fog Computing

Roberto Beraldi[1], Claudia Canali[2], Riccardo Lancellotti[2], Gabriele Proietti Mattia[1]
[1] Sapienza University of Rome, Italy [2] University of Modena and Reggio Emilia, Italy
[1] {beraldi, proiettimattia}@diag.uniroma1.it [2]{claudia.canali, riccardo.lancellotti}@unimore.it

ABSTRACT

Fog computing infrastructures must support increasingly complex applications where a large number of sensors send data to intermediate fog nodes for processing. As the load in such applications (as in the case of a smart cities scenario) is subject to significant fluctuations both over time and space, load balancing is a fundamental task. In this paper we study a fully distributed algorithm for load balancing based on random probing of the neighbors' status. A qualifying point of our study is considering the impact of delay during the probe phase and analyzing the impact of stale load information. We propose a theoretical model for the loss of correlation between actual load on a node and stale information arriving to the neighbors. Furthermore, we analyze through simulation the performance of the proposed algorithm considering a wide set of parameters and comparing it with an approach from the literature based on random walks. Our analysis points out under which conditions the proposed algorithm can outperform the alternatives.

KEYWORDS

Fog Computing; Load Balancing; Probe-based Algorithm

ACM Reference Format:
Roberto Beraldi[1], Claudia Canali[2], Riccardo Lancellotti[2], Gabriele Proietti Mattia[1]. 2020. Randomized Load Balancing under Loosely Correlated State Information in Fog Computing. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '20), November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3416010.3423244

1 INTRODUCTION

Fog computing is becoming a *de-facto* standard for the support of large distributed applications, such as smart cities applications, where data from a plethora of sensors is pre-processed, filtered and aggregated by intermediate fog nodes and is then sent to a cloud data center for additional analysis and storage. However, for the success of the fog computing vision, resource allocation is a key challenge due to finite resources at the fog level, increasing number and complexity of applications, and heterogeneity of incoming load due to mobile traffic [3]. In particular, achieving an adequate load balancing by distributing requests over the computing resources of the distributed fog infrastructure is a critical task for the support of the deployed applications. A fully centralized approach can achieve competitive performance [8], but has the weakness of high computational complexity and huge reporting overhead. Therefore, a centralized approach is not suitable for distributed fog computing systems: this consideration motivates our research on algorithms and protocols for a fully distributed approach to load sharing, without centralized components that act as data storage or orchestrator. Although resource sharing is a well-studied topic in the computer science community, some peculiar elements of fog computing do not fit the assumptions of studies in literature, such as: (1) the absence of a centralized entity that acts as a load balancer; (2) the heterogeneity among resource availability and local scheduling policies; (3) delayed state information among nodes.

The main contribution of this paper is a study of the impact of network latency on the effectiveness of randomization, which is the base of an important class of load balancing protocols, used to allocate on fog nodes jobs that are continuously generated from end devices (on-line load balancing). As a part of this analysis, we consider a power-of-random choices algorithm, namely *probe-based* where a fog node asks its neighbors' information on their load before taking forwarding decisions. This job offloading is regulated by a threshold Θ while workload information reflects the state of the probed node τ time units before the decision. We show that if τ is comparable with the service time, the load balancing performances deviate remarkably from the power-of-random choice's one and are similar to blind forwarding decisions. Furthermore, the probe-based algorithm is analyzed by means of simulation and compared with an existing alternative approach based on random walks, namely sequential forwarding. Our experiments highlight how the query fan-out (FO) parameter may play a complex role in the algorithm performance, including the possibility that a herding effect [4] may occur with a consequent deterioration of the performance. We also show that the probe-based algorithm may outperform the sequential forwarding alternative especially in a geographic scenario characterized by higher heterogeneity in terms of incoming load. Another important result concerns the impact on the performance of the FO parameter and of the herding effect, which is much less evident in a geographic heterogeneous scenario, thus identifying the proposed solution as the preferable choice for a realistic deployment scenario.

The rest of this paper is organized as following. Section 2 describes the proposed probe-based algorithms and the sequential forwarding algorithm. Section 3 presents a mathematical model of the system performance and Sec. 4 the evaluation of the proposed algorithm based on the Omnet++ simulation framework. Finally, Sec. 5 presents some final remarks.

2 ALGORITHM DEFINITION

We now introduce the *probe-based* load balancing algorithm that relies on a threshold to determine whether, upon receiving a new job, a probe for a less loaded neighbour is to be started. The threshold Θ is applied to the system load, that represents the number of jobs queued in the fog node (or being executed). This metric is used as an estimation of the waiting time for the incoming job. If a probe is to be started, the fog node issues query messages to a randomly selected set of FO neighbours. The parameter FO is the fan-out of the probe and ranges from $FO = 1$, meaning that only one random neighbour is selected, to $N - 1$ (N being the number of considered fog nodes), meaning that the probing involves every node in the infrastructure.

Algorithm 1 Probe-based Algorithm

```
Require: Θ, FO, Job
  if Job.IsForwarded() or System.Load() ≤ Θ then
    ProcessLocally(Job)
  else
    Neighs[] ← Random(System.neighbours(), FO)
    Responses[] ← ProbeNeighbours(Neighs[])
    BestNeigh ← SelectWithLowestLoad(Responses[])
    if System.Load() > BestNeigh.Load() then
      Forward(Job, BestNeigh)
    else
      ProcessOrDrop(Job, System.MaxQueue())
    end if
  end if
```

Algorithm 1 presents the formalization of the proposed algorithm. When a job from a sensor is received, the fog node uses the threshold Θ and the local load to decide if a probe for the neighbours' load should be issued (jobs forwarded from other fog nodes are processed locally without additional evaluation). If the probing is required, the fog node issues a set of query messages to the neighbours and waits for the response of every neighbour (this means that the higher is FO, the longer the probing phase takes). Each neighbour responds providing its load status, so the fog node can decide if the job should be forwarded to the neighbour with the lowest load or if the job is to be processed locally (if every neighbour has a higher load). It is worth to note that, in the case of high network delay (due slow network or due to network congestion), the load returned by a neighbour may be a *stale* information far different from the load the forwarded job will encounter. This may result in inaccurate forwarding decisions and can cause the so-called *herding effect*, known in literature [4]. A more detailed discussion on the effect of stale load information is provided in Sec. 3. When local processing occurs, the procedure *ProcessOrDrop*() is called: is there is space in the local queue, the job is enqueued for processing. If the queue is full, the job is dropped.

3 A MODEL FOR STALE INFORMATION

We consider $N \rightarrow \infty$ number of nodes, with capacity queue K receiving each a Poisson flow of job requests with rate λ and service time $\frac{1}{\mu} = 1$. The model assumes that the dynamic of these nodes is independent of each other and that the state information is available after a time interval τ. Let $X(t)$ be the queue length of a generic node in the system. From the Chapman-Kolmogorov, [5] equations, the probability transition functions of $X(t)$ can be expressed in a matrix form as:

$$\mathbf{P}(t) = e^{\mathbf{Q}t}$$

where:

$$P_{ij}(t) = Pr\{X(t+\tau) = j | X(0) = i\}$$

and $\mathbf{Q}$ is the standard $(K+1) \times (K+1)$ generation matrix of the process. Let's *suppose* that the process reaches the steady state, and let as usual $\pi_j = P_{ij}(\infty)$; also define $\tilde{\pi}_j = \sum_{i=j}^{K} \pi_i$. At the steady-state, the flow of jobs seen by node B when its state is j is:

$$\lambda_j = \begin{cases} \lambda + \lambda_j^F & \text{if } j \leq \Theta \\ \lambda \tilde{\pi}_j + \lambda_j^F & \text{otherwise} \end{cases} \tag{1}$$

where:

$$\lambda_j^F = \frac{1}{\pi_j} \lambda \sum_{i=0}^{K} \pi_i \tilde{\pi}'_{i+1} P_{ij}(\tau) \tag{2}$$

$$\tilde{\pi}'_i = \sum_{l=max\{i,\Theta+1\}}^{K} \pi_l \tag{3}$$

When the state is $j \leq \Theta$, see Equ. (1), the node in fact serves all the jobs coming from its users without any probe (this rate is λ), otherwise it serves jobs from its users if the probed has at least state j. The probed node replies with a state worst than j with probability $\tilde{\pi}_j$. In all states, it receives jobs from other nodes at rate λ_j^F. Equation (3) reflects the probabilities of the following events: (i) B sends a reply message to the probe message reporting state i, (ii) the state of A was at least $i+1$ – unless $i \leq \Theta$, in this case the state of A was at least $\Theta + 1$ since no probe messages are sent when the state is lower or equal to the threshold Θ; (iii) during τ time units the state of B changed from i to j. These probabilities are conditioned to the event of B being in state j.
A solution of the model is a matrix $\mathbf{Q}^*$ whose elements are tied by Equ. (1) and Equ. (3). $\mathbf{Q}^*$ is computed numerically using a Fixed Point algorithm: initially, a matrix $\mathbf{Q}_0$ is defined with $\lambda_i = \lambda$. Using the Equations (1) and (3) (where π is solution of $\pi_0 \mathbf{Q}_0 = 0$ and P_{ij} are the elements of its matrix exponential), a new generation matrix $\mathbf{Q}_1$ is then computed. From here, another matrix $\mathbf{Q}_2$ is derived in a similar way, and so on, until the maximum difference among any two elements of the successive matrix is less than an error $\epsilon = 10^{-13}$.

Two extreme cases are now considered: (i) state information is updated, i.e., $\tau = 0$; (ii) completely uncorrelated states. In the first case $\mathbf{P}(0) = \mathbf{I}$, i.e.:

$$P_{ij}(0) = \delta_{ij}$$

The state of the probed node cannot change w.r.t. to the reported value, i.e., no state transition occurs. By substituting this value in Equ. (3) we get:

$$\lambda_j^F = \lambda \frac{1}{\pi_j} \sum_{i=0}^{K} \tilde{\pi}_{i+1} \pi_i \delta_{ij} = \lambda \frac{1}{\pi_j} \pi_j \tilde{\pi}'_{j+1} = \lambda \tilde{\pi}'_{j+1}$$

so that:

$$\lambda_j = \lambda \begin{cases} 1 + \tilde{\pi}_{\Theta+1} & \text{if } j \leq \Theta \\ \tilde{\pi}_j + \tilde{\pi}_{j+1} & \text{otherwise} \end{cases} \tag{4}$$

For $\Theta = 0$ the above equations describe the dynamic of the power of two random choices algorithm on a loss queue model [7]. In fact, the generic balance equation of the *MC* associated to $\mathbf{Q}$ becomes (recall $\mu = 1$):

$$\lambda(\tilde{\pi}_j + \tilde{\pi}_{j+1})\pi_j = \pi_{j+1}$$

Since $(\tilde{\pi}_j + \tilde{\pi}_{j+1})(\tilde{\pi}_j - \tilde{\pi}_{j+1}) = (\tilde{\pi}_j^2 - \tilde{\pi}_{j+1}^2)$ and $\pi_{j+1} = \tilde{\pi}_{j+1} - \tilde{\pi}_{j+2}$ the equations can be rewritten in the so-called supermarket fluid model form (where conventionally $\tilde{\pi}_{K+1} = 0$) [6]:

$$\lambda(\tilde{\pi}_{j-1}^2 - \tilde{\pi}_j^2) = \tilde{\pi}_j - \tilde{\pi}_{j+1}$$

These equations have the following fluid flow interpretation. In a population of $N \longrightarrow \infty$ nodes, any node sends its jobs to a central scheduler. The scheduler then sends the jobs to the least loaded among two random nodes. Here $\tilde{\pi}_j$ is interpreted as the *fraction* of nodes with at least j jobs en-queued.

The second case corresponds to the ideal case of $\tau \longrightarrow \infty$, because this ensures to observe the node in two random steady states.

$$P_{ij}(\infty) = \pi_j$$

hence:

$$\lambda_j^F = \frac{1}{\pi_j}\lambda \sum_{i=0}^{K} \tilde{\pi}'_{i+1}\pi_i\pi_j = \lambda \sum_{i=0}^{K} \tilde{\pi}'_{i+1}\pi_i = \lambda \sum_{i=0}^{\Theta} \tilde{\pi}_{\Theta+1}\pi_i + \lambda \sum_{i=\Theta+1}^{K} \tilde{\pi}_{i+1}\pi_i$$

so that:

$$\lambda_j = \begin{cases} \lambda + \lambda_j^F & \text{if } j \leq \Theta \\ \lambda\tilde{\pi}_j + \lambda_j^F & \text{otherwise} \end{cases} \tag{5}$$

The above equation can also be interpreted as following. When the state is above the threshold, a node serves the job with probability $\tilde{\pi}_j$ otherwise it forwards the job to a random node. As π_j is same for all nodes, the decision to forward can be taken by looking at the workload history of the node itself (for example, the node can estimate its occupancy probability distribution over time). A node at state j then forwards a job with the same probability that the remote node is less loaded than itself, i.e., $1 - \tilde{\pi}_j$. When even this info is not used, the algorithm makes blind decisions.

3.1 Performance Metrics

3.1.1 *Dropping probability.*

The probability to drop a job for the query-based algorithm is:

$$p_B = \pi_K^2 + \sum_{i=\Theta}^{K-1} \tilde{\pi}_{i+1}\pi_i P_{iK}(\tau) \tag{6}$$

because a job is dropped if: (i) the receiving node is full but the job cannot be forwarded since the receiving node reports it is full as well (first term), or (ii) the job is forwarded, but during the time τ the target node becomes full (the job finds the node in state K) and drops the job. For the blind one:

$$p_B = \tilde{\pi}_{\Theta+1}\pi_K \tag{7}$$

which reflects the fact that a job is forwarded towards a congested node, i.e. whose state is K.

3.1.2 *Control overhead.*

The control overhead is given by the probe message rate:

$$c_{ovh} = \lambda \sum_{i=\Theta+1}^{K} \pi_i = \lambda\tilde{\pi}_{\Theta+1} \tag{8}$$

3.2 Numerical results

We now provide some numerical results obtained for $K = 10$, for different loads and delays. The service time is $\mu = 1$.

Load balancing under perfect knowledge. This case represents the best-case scenario, where updated state information are immediately available to all nodes. The state of the probed node is always the same of the reported one, $\tau = 0$ complete correlation. To limit the control information overhead nodes set Θ to some proper value (Fig. 1).

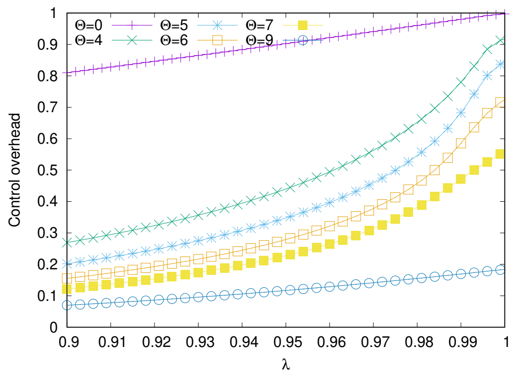

Figure 1: Control overhead vs loads.

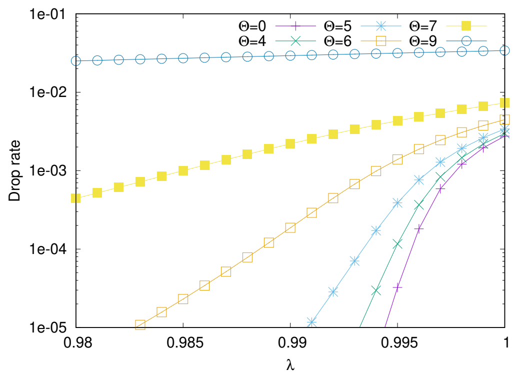

Figure 2: Drop rate performance vs loads $\tau = 0$.

Figure 2 shows the drop rate as a function of the load λ ($\mu = 1$) and different T. The threshold is, in this case, an effective way to reduce the control overhead. With $\Theta = 4$ the load balancing reaches almost the same drop rate of $\Theta = 0$ but with much less overhead (about 1/3 for $\lambda < 0.9$).

Load balancing under partial knowledge. While threshold is a simple way to reduce the control overhead, the amount of control information can still remain a source of latency so that state pulled by the probe messages may arrive after some not negligible delay. Figure 3 shows how τ affects the drop rate. Simulation results for $N = 3000$ nodes obtained via a custom python simulator are also shown. The simulated model is very simple as it makes available state information older of τ time units w.r.t. the current simulated time, while job transmission is instantaneous. For $\lambda = 0.95$ and $\tau = 0$ no losses were observed. These results provide evidence that the model captures the key behaviour of the algorithm. A more detailed and realistic simulation study is reported later in the paper.

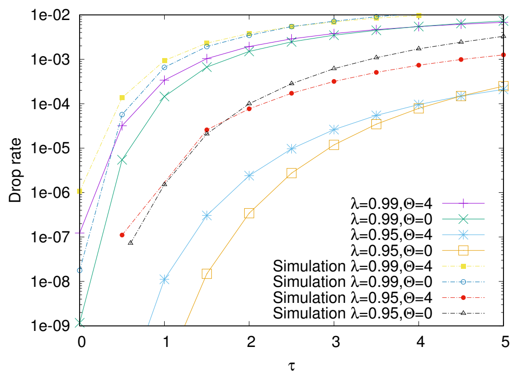

Figure 3: Effect of τ on the performance.

For a load as high as $\lambda = 0.99$, the drop rate increases considerably as soon as $\tau > 0$. For example for $\tau = 0.5$, i.e., a half of the service time, the drop rate increases from 1.17×10^{-9} to 5.48×10^{-06} and for $\tau = 1$ it reaches 1.4×10^{-04}, i.e, about 5 order of magnitude higher than when correct state information are immediately available. As τ increases, the drop rate for $\Theta = 4$, is lower than for $\Theta = 0$ because the algorithm makes a lower number of wrong decisions, e.g., a node forwarding the job to another node that becomes full after τ time units from when it had some free place in the queue. Overall, we then expect that in a real setting even small delays in getting information will weaken the power-of-random choice effect of a real system remarkably.

4 SIMULATION RESULTS

To evaluate the performance of the proposed load balancing algorithm, we rely on a discrete event simulator based on Omnet++. Specifically, we aim to capture the impact of the network delay (and congestion) on the load balancing effectiveness. To this aim, we compare the proposed probe-based algorithm and the sequential forwarding algorithm from literature [1].

For the experiments we focus on a more complex setup derived from a realistic topology based on an ongoing project of traffic sensing in Modena, a city in northern Italy of roughly 180'000 inhabitants. The sensors, located in the main city streets, collect information about traffic and air quality. Fog nodes, placed in municipality buildings,exchange data using long-range wireless links (such as IEEE 802.11ah/802.11af) to interact with the sensors and among themselves. The scenario description is generated using the PAFFI framework [2]. As in these links the available bandwidth decreases with distance, we assume the bandwidth to be inversely proportional to the distance between the two communication endpoints. We choose bandwidth and job size so that the time to transfer a job is comparable to the processing time ($\delta = 1/mu = 10$ms). Each sensor communicates with the closest fog node: hence, the incoming load on each fog nodes is highly heterogeneous (from 2× the processing capacity to the case of a highly underutilized fog node). The average system utilization is $\rho = \overline{\lambda}/rho = 0.9$. Finally, we set the queue size in fog nodes to 10 jobs as in [1].

In our experiments we consider two main performance metrics: (1) *Drop rate*, that is the probability of a job being discarded due to the full queue of the selected fog node; (2) *Response time*, that is the time between the arrive of the Job at the first fog node and the end of processing on the final fog node. We also consider the breakdown of the response time in its following components: *Service time* that is the time spent being processed; *Balancer time* that is the time spent being forwarded among the fog nodes (as previously said this contribution can represent the τ parameter in the model of Sec. 3); *Queuing time* that is the time spent in the fog node ready queue waiting to be processed. Furthermore, we consider the following parameters to describe each simulation scenario: (1) Θ is the threshold that triggers the cooperation (in our experiments $\Theta \in [1, 10]$, where 10 is the maximum queue length); (2) *FO* is the fan-out of a probe, that is the number of neighbors to which the load query message is sent ($FO \in [1, N-1]$, where $N = 20$ is the number of fog nodes in our simulation).

As pointed out in the theoretical model of Sec. 3, we anticipate the impact of network delays on the effectiveness of the probe-based algorithm, and the role that the *FO* parameter may play in the algorithm performance. Indeed, as *FO* grows, we have a three-fold effect: (1) we increase the ability of a probe to identify the lowest loaded neighbor; (2) we increase the delay to complete the probe due to higher network utilization, resulting in higher probability of having a stale (and inaccurate) load information; (3) as more nodes may receive multiple queries in a short amount of time, we may have a *herding* effect [4] (when many fog nodes forward their job to the same less-loaded neighbor, causing its overload).

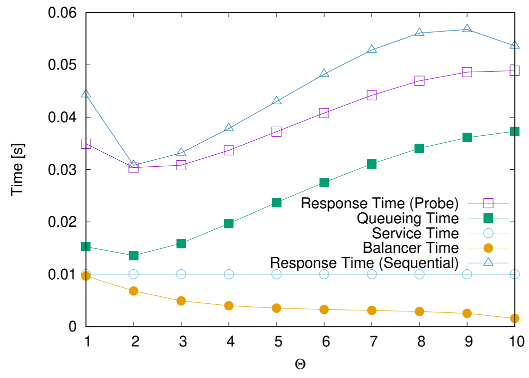

Figure 4: Response time *vs.* Threshold Θ ($FO = 3$)

Fig. 4 provides a comparison of the response time achieved by the probe-based and sequential forwarding algorithms when $FO = 3$. Furthermore, a breakdown of the contribution of the response time of the probe-based algorithm is provided.

Comparing the two algorithms we observe that the probe-based algorithm provides a performance gain in terms of response time against the sequential forwarding alternative. Furthermore, from the breakdown of the response time we observe that the time spent looking for a suitable neighbor (balancer time) decreases with Θ, for the two-fold reason that (1) the balancing function is activated with a lower frequency and (2) the lower network utilization (due to the lower number of probes) results in faster query-response during the probe phase. On the other hand, as we activate less often the balancer, we accept to process the jobs on local node with a potentially higher load, as testified by the higher queuing time.

Having described the basic behavior of our load balancing algorithm we now focus on the impact of the main parameters that may affect its performance. We start discussing the impact of the query fan-out *FO*.

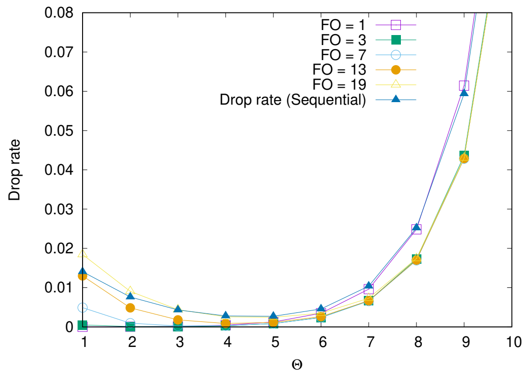

Figure 5: Drop rate for different Fan-out *FO*

Fig. 5 shows the drop rate *vs.* the threshold Θ for different values of *FO*. We observe that the drop rate follows a cup-shaped curve. The drop rate grows when the threshold is too low due to load balancing strategy activated even when not necessary, and interfering with the overall ability to find a suitable neighbor in case of actual overload. In a similar way, the drop rate increases when the threshold is too high, because the load balancing is not activated even in case of overload. As *FO* grows, the curve of the drop rate become more accentuated. This means that when *FO* is high, the load returned by the probing phase is uncorrelated with the load found on the node when the job is forwarded. This effect, described in the theoretical model for high values of τ has a twofold explanation. First, as *FO* grows, the probe takes longer to complete due to the higher network utilization. As a consequence the load on the neighbor node have more time to evolve drifting away from the value provided when answering the query. Second, the *herding effect* [4] may occur so that, when a node reports a load lower than the average, several neighbors will select this node as the target for job forwarding, causing overload. In the worst cases the curve assumes the shape close to the sequential forwarding algorithm [1], when a job is sent to a randomly selected neighbor.

Focusing on Fig. 6, with the response time for different values of *FO*, we observe that the probe-based algorithm is generally faster compared with the sequential forwarding algorithm with the same threshold value. Especially for high values of the threshold Θ, higher values of fan-out provide a benefit in terms of response time. Indeed as Θ grows we have less queries and this reduces the impact of the herding effect and makes the probing mechanism more effective. A further confirmation of the interaction of the number of queries with the insurgence of herding effect that hinder the cooperation effectiveness is provided by the observation that, as *FO* increases,

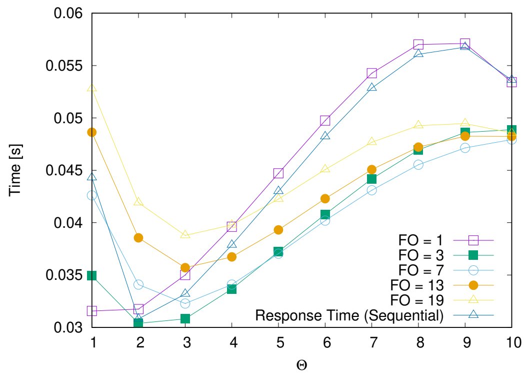

Figure 6: Response time for different Fan-out *FO*

the threshold value that provides the best response time increases from Θ = 2 to Θ = 3, that is towards a case where the number of queries issued is lower.

5 CONCLUSIONS

In this paper we focus on the load balancing issue of distributing incoming jobs over the nodes of a fog computing infrastructure. Our proposal is designed to be fully decentralized, and aims to be a viable solution for a realistic fog deployment in a smart city scenario possibly characterized by heterogeneous workload distribution. We analyze the impact of network latency on the effectiveness of the selection of fog nodes to allocate the incoming jobs, proposing a probe-based algorithm for load balancing. We evaluate the performance of our proposal using both a mathematical model and a simulator. Our analysis revealed that taking schedule decisions based on state information received even with a small delay compared to the service time reduces the load balancing effectiveness considerably. To take this effect into account, we studied a threshold probe-based algorithm with small fan-out which is a preferable choice with respective to the existing alternative especially in a geographic realistic scenario characterized by a higher level of heterogeneity in terms of incoming load.

REFERENCES

[1] R. Beraldi, C. Canali, R. Lancellotti, and G. Proietti Mattia. 2020. A random walk based load balancing algorithm for Fog Computing. In *The Fifth International Conference on Fog and Mobile Edge Computing (FMEC 2020)*. Paris, France, 1–8.
[2] C. Canali and R. Lancellotti. 2019. PAFFI: Performance Analysis Framework for Fog Infrastructures in realistic scenarios. In *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. Rome, Italy, 1–8.
[3] S. Chen, T. Zhang, and W. Shi. 2017. Fog Computing. *IEEE Internet Computing* 21, 2 (2017), 4–6.
[4] M. Dahlin. 2000. Interpreting Stale Load Information. *IEEE Transactions on Parallel and Distributed Systems* 11, 10 (oct 2000), 1033–1047.
[5] Eli Upfal Michael Mitzenmacher. 2005. *Probability and computing: randomized algorithms and probabilistic analysis*. CAMBRIDGE UNIVERSITY PRESS, The Edinburgh Building, Cambridge CB2 2RU, UK.
[6] Andrea W Richa, M Mitzenmacher, and R Sitaraman. 2001. The power of two random choices: A survey of techniques and results. *Combinatorial Optimization* 9 (2001), 255–304.
[7] Qiaomin Xie, Xiaobo Dong, Yi Lu, and Rayadurgam Srikant. 2015. Power of d choices for large-scale bin packing: A loss model. *ACM SIGMETRICS Performance Evaluation Review* 43, 1 (2015), 321–334.
[8] A. Yousefpour, G. Ishigaki, and J. P. Jue. 2017. Fog Computing: Towards Minimizing Delay in the Internet of Things. In *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE, Piscataway, New Jersey, US, 17–24.