Randomized load balancing under loosely correlated state information in fog computing

Roberto Beraldi
Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome
Rome, Italy
beraldi@diag.uniroma1.it

Claudia Canali
Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia
Modena, Italy
claudia.canali@unimore.it

Riccardo Lancellotti
Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia
Modena, Italy
riccardo.lancellotti@unimore.it

Gabriele Proietti Mattia
Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome
Rome, Italy
proiettimattia@diag.uniroma1.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, it is fundamental to provide load balancing across the infrastructure. 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 that we take into account the impact of delay during the probe phase and we show how the presence of stale information impact on the algorithm performance. We propose a theoretical model for the loss of correlation between actual load on a node and stale information arriving to the neighbors with a network-induced delay. Furthermore, we analyze through simulation the performance of the proposed algorithm considering a wide set of parameters and comparing it with an alternative approach from the literature that is 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, Simulation

ACM Reference Format:
Roberto Beraldi, Claudia Canali, Riccardo Lancellotti, and Gabriele Proietti Mattia. 2020. Randomized load balancing under loosely correlated state information in fog computing. In *MSWiM '20: The 23rd International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

MSWiM '20, November 16–20, 2020, Alicante, Spain

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

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. The fog vision is a clear step ahead compared to a traditional cloud-only approach because it can provide lower response times (because latency-bound tasks can be placed on the network edge at the level of fog nodes) and better utilization of the available network resources (thanks to the data filtering and aggregation on the fog nodes) [12, 15]. 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 [5]. 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 [18], but it has the weakness of high computational complexity and huge reporting overhead. Therefore, the centralized approach is not suitable for distributed fog computing systems. This consideration motivates our research on algorithms and protocols to support a fully distributed approach to load sharing, without any need for a centralized component that act as data storage or orchestrator. Although resource sharing is a classical and well-studied topic in the computer science community, the model of fog computing does not fit all the assumptions of the studies available in the literature. In particular, the following elements are peculiar to the fog deployment: (i) the absence of a centralized entity that acts as a load balancer; (ii) the heterogeneity among resource availability and local scheduling policies; (iii) 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. For our evaluation, we rely on a discrete event simulator developed starting from the Omnet++ framework and we consider a wide set of parameters over two main scenarios: a simplified one, namely uniform mesh scenario, and a more complex geographic scenario derived from a realistic topology based on a medium-size city in Italy. Our experimental evaluation highlights how the query fan-out (*FO*) parameter may play a complex role in the algorithm performance, including the possibility that a herding effect [6] may occur with a consequent deterioration of the performance. Our experiments also show that the probe-based algorithm may outperform the sequential forwarding alternative especially in the geographic scenario characterized by a higher level of 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 discusses related works. Section 3 describes the proposed probe-based algorithms and the sequential forwarding algorithm used as a term of comparison. Section 4 presents a mathematical model to describe system performance. In Sec. 5 we provide an evaluation of the proposed load balancing algorithm based on the Omnet++ simulation framework over two different scenarios. Finally, Sec. 6 presents the conclusions and future research directions.

2 RELATED WORK

In this section we briefly analyze the state of the art regarding load distribution and load balancing algorithms in the context of fog computing systems.

In [13], an algorithm called Multi-tenant Load Distribution Algorithm for Fog Environments (MtLDF) has been proposed to optimize load balancing in Fogs environments considering specific multi-tenancy requirements. However, the proposed load balancing scheme adopts a centralized fog management layer that receives all the state information about the fog nodes. On the other hand, our solution relies on a fully distributed approach.

In [8], the incoming jobs consist of tasks that are characterized according to their computational nature and, based on this classification, they are allocated to the appropriate node. Edge networks communicate through a brokering system with IoT systems in an asynchronous way via the Pub/Sub messaging pattern. However, again the approach relies on a centralized component, that is a workload balancer, which is required by the solution.

In [18], an approach similar to the sequential forwarding algorithm is used. However, the proposed solution requires either a centralized repository to store the load state of each fog node or needs a specific protocol to send updates on the load state of each node. Our approach, exploiting a probe-based approach, provides good performance without requiring coordination structures involving all nodes but reducing the information exchange to a limited set of nodes.

The solution presented in [16] is based on the periodical distribution of the incoming tasks in the edge computing network in order to increase the number of tasks that can be processed at the edge of the network and at the same time to satisfy the quality-of-service (QoS) requirements of the incoming tasks. The model, however, assumes the availability of a batch of tasks to be assigned, i.e., the tasks are not processed online, as it typically occurs in smart cities applications considered in our approach.

The studies in [1, 2] adapted the idea of randomly select nodes to offload a task used in the class of power-of-choices algorithms to the fog computing model. The Sequential Forwarding algorithm, based on the random node selection and used as a comparison for the performance evaluation in this paper, was proposed by the same authors, in [3]. The proposal of a probe-based algorithm represents a clear step ahead with respect to the blind approach of the Sequential Forwarding algorithm, both in terms of performance and advantage in the case of a (more realistic) heterogeneous fog infrastructure.

Finally, in [6] the problem of managing stale load information and mitigating herding effect in Web-based systems is analyzed. Our approach analyzes these effects in a different fog-related scenario, pointing out the similarities and providing a more general theoretical framework for this type of effects.

3 ALGORITHMS DEFINITION

We now introduce the load balancing algorithm proposed in our research to support workload management in a distributed fog computing infrastructure. Specifically, we first introduce the proposed *probe-based* algorithm and we present also a load-blind algorithm already presented in literature [3] that is used as a comparison in the performance evaluation.

3.1 Probe-based algorithm

Figure 1 depicts a model of the *probe-based* algorithm outlining its main functional components including probe messages, job forwarding and node system load status. Furthermore the figure outlines the presence of network delays.

In detail, the algorithm relies on a threshold to determine whether, upon receiving a new job, a probe for a less loaded neighbor 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* neighbors. The parameter *FO* is the fan-out of the probe and ranges from $FO = 1$,

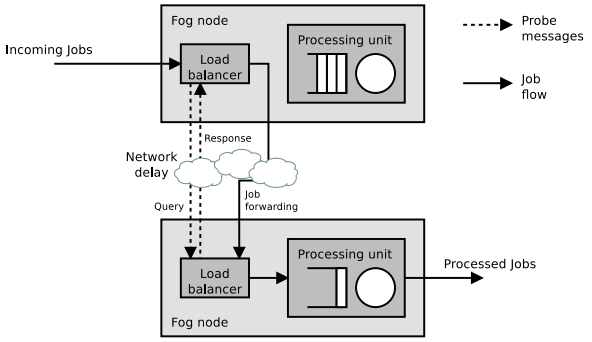


Figure 1: Probe-based load balancing algorithm

meaning that only one random neighbor 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*() $\leq \Theta$ **then**
    ProcessLocally(Job)
  else
    Neighs[] $\leftarrow$ *Random*(System.*Neighbors*(), *FO*)
    Responses[] $\leftarrow$ *ProbeNeighbors*(Neighs[])
    BestNeigh $\leftarrow$ *SelectWithLowestLoad*(Responses[])
    if System.*Load*() > BestNeigh.*Load*() **then**
      Forward(Job, BestNeigh)
    else
      ProcessLocally(Job)
    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 neighbors' 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 neighbors and waits for the response of every neighbor (this means that the higher is *FO* the longer is likely this probing phase to take). Each neighbor provides within the response its load status, so the fog node can decide if the job should be forwarded to the neighbor with the lowest load or if the job is to be processed locally (if every neighbor 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 neighbor 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 [6]. A more detailed discussion on the effect of stale load information is provided in Sec. 4.

Algorithm 2 details the case where a node is processed locally (for example, due to the call to the *ProcessLocally*() procedure in Alg. 1). In this case, the Job should be enqueued in the ready queue

Algorithm 2 Local processing: *ProcessLocally*()

Require: Job
  if System.*Queue*() < System.*MaxQueue*() **then**
    Enqueue(Job)
  else
    Drop(Job)
  end if

of the server. However, since this queue is finite in size, if the queue is already full, the job is dropped, resulting in a loss.

3.2 Sequential forwarding algorithm

The *Sequential Forwarding* algorithm uses a threshold Θ to decide if an incoming job must be forwarded to a neighbor or not, just like the previous probe-based algorithm. Furthermore, the algorithm relies on an additional parameter M (that is the a maximum number of steps), in order to guarantee a bound on the delay experienced by each job during the load balancing phase. A detailed description of the sequential forwarding algorithm is provided in [3]. In this paper we use this algorithm as a comparison for our proposal and we set the parameter $M = 5$ that is a value proved in preliminary experiments to provide low response time and low drop rate.

Algorithm 3 Sequential Forwarding Algorithm

Require: M, Θ, Job
  if Job.*Steps*() $\geq M$ **then**
    ProcessLocally(Job)
  else
    if System.Load() $\leq \Theta$ **then**
      ProcessLocally(Job)
    else
      Neigh $\leftarrow$ *Random*(System.*Neighbors*(), 1)
      Job.*IncrementSteps*()
      Forward(Job, Neigh)
    end if
  end if

The sequential forwarding process is detailed in Alg. 3. We assume that the data structure describing the job is enriched with meta-data to keep track of the number of times a job is forwarded.

4 A MODEL FOR STALE INFORMATION

The probe-based algorithm relies on state information gathered from the other nodes. If the duration of the scheduled work is comparable with the latency required to get this information, the reported state maybe out-of-date. We want to study how taking scheduling decisions on information that does not reflect the actual state of the selected node may affect the effectiveness of the load balancing protocol. This problem has been studied in seminal paper [11], but in a different interaction model. To put the problem in a broader context, we assume that the time interval from when a node reports a state until a job arrives to that node is a constant value τ. Although the propagation delay of the backhaul can be small, this time can be dominated by the packet processing time.

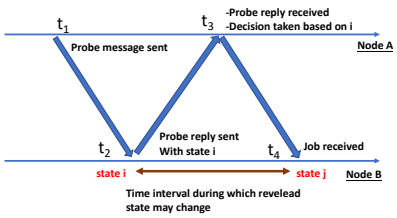


Figure 2: Time diagram concerning the probe message

Figure 2 reports a typical interaction pattern of our query-based protocol. At time t_1, node A receives a job and sends the probe message; at time t_2 B receives the probe message, reads its state and sends the reply to A; A receives the reply message at time t_3, compares its state with the one reported by B and takes its forwarding decision based on the reported state; at time t_4 B receives the job. In this example, $\tau = t_4 - t_2$ and if τ is long enough the state of B is in general $j \neq i$. A similar update pattern can arise when probes are pipelined, i.e., node A uses the first probe reply available even if triggered by a previously received job. Other strategies (not investigated in this work), may be based on proactive periodic state information broadcast to random groups.

4.1 Loss model

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 are independent from each other. Let $X(t)$ be the queue length of a generic node in the system. From the Chapman-Kolmogorov, [10] 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 $(K+1) \times (K+1)$ generation matrix of the process.

$$\mathbf{Q} = \begin{bmatrix} -\lambda_0 & \lambda_0 & 0 & \dots & 0 \\ 1 & -1-\lambda_1 & \lambda_1 & \dots & 0 \\ 0 & 1 & -1-\lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -1 \end{bmatrix}$$

The entry λ_j is the transition rate from j to $j+1$.

4.1.1 Flow rates. 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:

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

where:

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

and:

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

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 (2) 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.

4.1.2 Numerical solution. A solution of the model is a matrix $\mathbf{Q}^*$ whose elements are tied by Equ. (1) and Equ. (2). $\mathbf{Q}^*$ 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 (2) (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 $\epsilon = 10^{-13}$.

4.1.3 Completely correlated states. This condition occurs when communication is instantaneous , $\tau = 0$. In this case $\mathbf{P}(0) = \mathbf{I}$, i.e.:

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

The state of the probed node cannot change wrt to the reported value, i.e., no state transition occurs. By substituting this value in Equ. (2) 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:

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

For $\Theta = 0$ the above equations describe the dynamic of the power of two random choices algorithm on a loss queue model [17]. 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 $\tilde{\pi}_{K+1} = 0$) [14]:

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

These equations have the following fluid flow interpretation. In a population of $N \rightarrow \infty$ 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.

4.1.4 Completely uncorrelated states. This case ideally corresponds to $\tau \to \infty$ 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:

$$\lambda_j = \begin{cases} \lambda + \lambda_j^F & \text{if } j \le \Theta \\ \lambda\tilde{\pi}_j + \lambda_j^F & \text{otherwise} \end{cases} \quad (5)$$

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.

4.1.5 Blind forwarding. In sequential forwarding nodes send job without any probe. For $M = 1$ we call this protocol blind (as no state info are used to decide). A node never serves the job when it is above Θ (contrary from the previous completely uncorrelated case where it serves locally with probability $\tilde{\pi}_j$). The rates are:

$$\lambda_j = \lambda \begin{cases} 1 + \tilde{\pi}_{\Theta+1} & \text{if } j \le \Theta \\ \tilde{\pi}_{\Theta+1} & \text{otherwise} \end{cases} \quad (6)$$

which reflects the fact that the job arrivals at the selected node are uncorrelated from its state. The formula is easily extended to any M.

4.2 Performance Metrics

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

$$p_B = \pi_K^2 + \sum_{i=\Theta}^{K-1} \tilde{\pi}_{i+1}\pi_i P_{iK}(\tau) \quad (7)$$

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:

$$p_B = \tilde{\pi}_{\Theta+1}\pi_K \quad (8)$$

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

4.2.2 Control overhead. The control overhead is given by the probe message rate:

$$c_{ovh} = \lambda \sum_{i=\Theta+1}^{K} \pi_i = \lambda\tilde{\pi}_{\Theta+1} \quad (9)$$

4.3 Numerical results

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

4.4 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. 3). Figure 4 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 $\lambda < 0.9$).

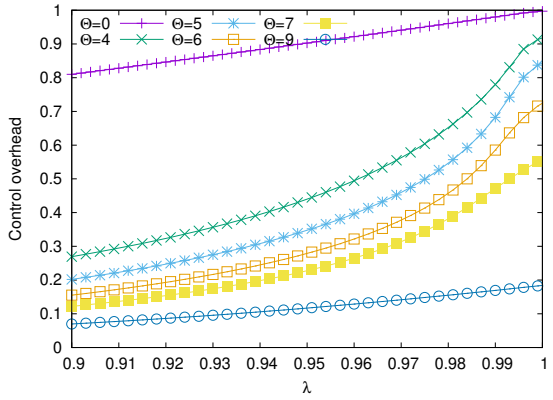


Figure 3: Control overhead vs loads.

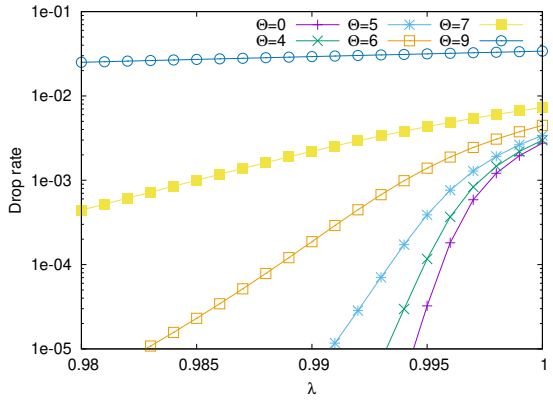


Figure 4: Drop rate performance vs loads $\tau = 0$.

4.5 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 5 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 wrt 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.

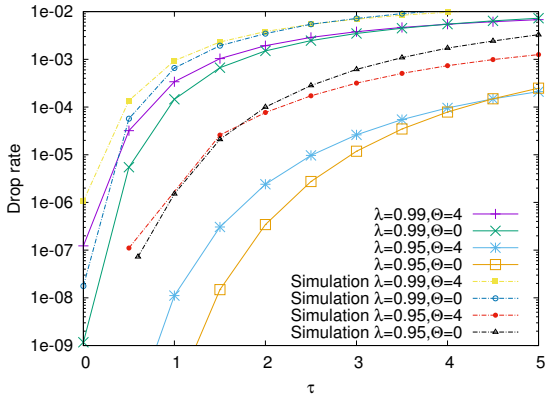


Figure 5: Effect of τ on the performance.

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.

4.6 Blind Load balancing

In blind forwarding, nodes send a job without any info about the state of the remote node. This extreme case avoids the need of any control information. Figure 6 shows the drop rate for different thresholds. The horizontal line corresponds to $\tau = 30$ and no load balancing. The drop rate for the uncorrelated case is lower than the blind meaning that the state information received from the probe indirectly reveals the average (expected) state of the other node.

5 SIMULATION RESULTS

To evaluate the performance of the proposed load balancing algorithm, we rely on a discrete event simulator based on the Omnet++ framework[1], with additional modules developed *ad-hoc* to support the algorithms considered in our experimental evaluation.

In our analysis we take advantage from the insight provided by the model described in Sec. 4 and we aim to capture the impact of

[1]https://omnetpp.org/

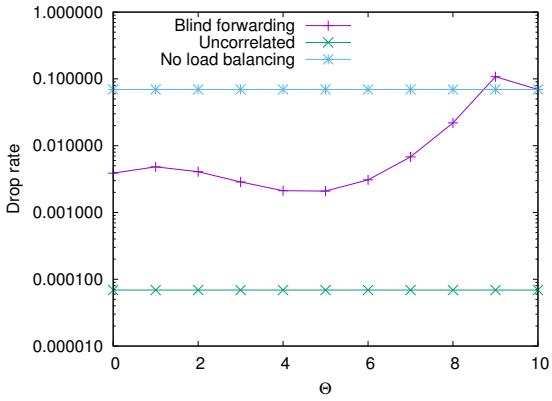


Figure 6: Drop probability blind algorithm

the impact of the network delay on the load balancing effectiveness. To this aim we model the network taking into account the network bandwidth available between each couple of fog nodes and we consider the possibility of network congestion, when multiple probe messages and jobs being forwarded must queue to access the network link. In our experiments, the time spent on average during the probing and forwarding (we anticipate that we will refer to this time as *balancer time*) plays the role of the parameter τ in the model.

5.1 Experimental setup

In our experiments we consider both the proposed probe-based algorithm and the sequential forwarding algorithm from literature [3], as a comparison. Furthermore, we consider two different scenarios, namely *uniform mesh scenario* and *geographic scenario*.

In the *uniform mesh scenario* we consider a simplified setup where all nodes are identical and can be described as M/M/1/K queues, with a maximum length of $K = 10$ elements, with the same incoming load $\lambda = 0.9$ jobs/sec, the same processing rate $\mu = 1$ job/sec (that result in an overall utilization $\rho = 0.9$ of the infrastructure). Furthermore, the bandwidth is the same for all the connections among the fog nodes. In particular, the bandwidth BW is 8Mbit/s and we assume the job size S to be 1Mbyte so that the delay to forward one job from one fog node to another is 1 sec (these values are compatible with the use of long-range, low-power wireless links for fog-to-fog communication [9]). The considered setup ensures that the communication time ($\delta_J = S/BW$) is comparable to the processing time ($1/\mu$), that is a qualifying point in the fog scenario, as pointed out in Sec. 1. Furthermore, we model also the bandwidth requirements of probe messages, defining δ_Q as the time required to send and receive a probe message (not considering the queuing time at the network level due to congestion in the underlying links). We recall that our simulation can capture the arising on congestion on the network when multiple probes and jobs forwarded must compete for the available bandwidth.

In the *geographic scenario*, instead, 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 are located in the main city streets

and collect information about the traffic (taking pictures of the street when movement is sensed to count how many cars are passing). These data are integrated with data on the air quality. Fog nodes are placed in facilities belonging to the municipality and exchange data using long-range wireless links (such as IEEE 802.11ah/802.11af [9]) to interact both with the sensors and among themselves. the scenario description is generated using the PAFFI framework [4]. As in these links the available bandwidth decreases withe distance, we assume the bandwidth to be inversely proportional to the distance between the two communication endpoints. Each sensor communicate with the closest fog node as in [7] so that the incoming load on each fog node is highly heterogeneous, ranging from cases where the incoming load is more than 2× the processing capacity to cases where a fog node is highly underutilized.

Throughout our experiments we consider the following main performance metrics:

- *Drop rate*, that is the probability of a job being discarded because the queue of the selected fog node is completely full.
- *Response time*, that is the time occurring between the moment the Job is received from the first fog node, to the moment the processing ends on the final fog node.

We consider also useful, when considering the response time, to provide its breakdown in the 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 pointed out this contribution can play the role of the τ parameter in the model of Sec. 4),
- *Queuing time* that is the time spent in the fog node ready queue waiting to be processed.

Furthermore, in our simulation evaluation, we consider the following parameters to describe each simulation scenario:

- Θ is the threshold used to decide if the cooperation is to be triggered. In our experiments $\Theta \in [1, 10]$, where 10 is the maximum queue length.
- FO is the fan-out of a probe, that is the number of neighbors to which the load query message is sent. In our experiments we consider $FO \in [1, N-1]$, where $N = 20$ is the number of fog nodes in our simulation.
- $\eta_Q = \delta_Q/\delta_J$ is the impact of the cooperation delay (that is the time for sending a query and receiving the answer) compared to the job forwarding delay. We can think of η_Q as the ratio between the size of a message used for cooperation and a message containing a job. In our experiments we consider $\eta_Q \in [1\%, 100\%]$

As pointed out in the theoretical model of Sec. 4, we anticipate the critical impact of network delays on the effectiveness of the probe-based algorithm. Furthermore, we point out that the FO parameter may play a complex role in the algorithm performance. Indeed, as FO grows, we have a three-fold effect. On one hand, we increase the ability of a probe to identify the lowest loaded neighbor. On the other hand, we increase the delay to complete the probe, as the higher network results in longer network delays and we have to wait for the slowest node to respond; this higher network delay in turn increases the probability of having a stale load information from a probe that no longer represent the load on the remote nodes. Finally, as more nodes may receive multiple queries in a short amount of time, we may have a *herding* effect [6], that is the case where many fog nodes forward their job (due to stale load data) to the same less-loaded neighbor, causing its overload.

5.2 Uniform mesh scenario

We start our analysis focusing on the Uniform mesh scenario.

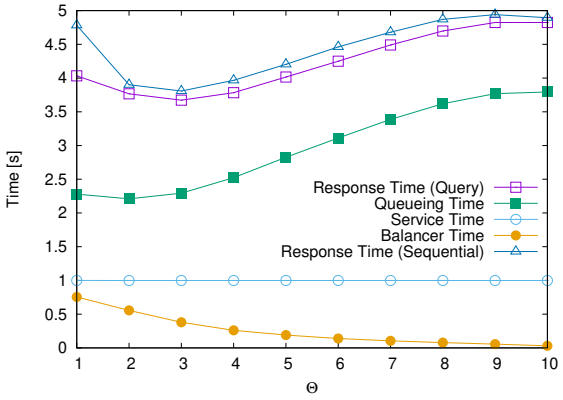


Figure 7: Response time *vs.* Threshold Θ ($FO = 3$, $\eta_Q = 10\%$)

Figure 7 shows the response time of the proposed probe-based algorithm compared to the sequential forwarding alternative for several values of the threshold Θ. Furthermore, a breakdown of the response time is provided. Our results are related to the scenario where $FO = 3$, $\eta_Q = 10\%$, but our main findings have general validity. Comparing the two algorithms we observe that the probe-based algorithm provides a (small) performance gain in terms of response time against the sequential forwarding alternative. Furthermore, from the breakdown of the response time we observe that, while the service time remains constant with respect to the threshold, the time spent looking for a suitable neighbor (that is the 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 increase of the queuing time.

Figure 8 provides a further proof of the impact of the delay associated with the load balancing on the drop rate of the algorithm. In particular we focus on the range where $\Theta \leq 5$, that is the the range where the load balancing is activated more often and where this effect is more evident. We observe how an higher delay results in an higher drop rate as suggested by the theoretical model.

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. Specifically, Fig. 9 shows the drop rate (Fig. 9a) and the response time (Fig. 9b) as a function of FO for the case where $\eta_Q = 10\%$. Again results for other values of η_Q are omitted for space reasons and because they do not provide additional insight.

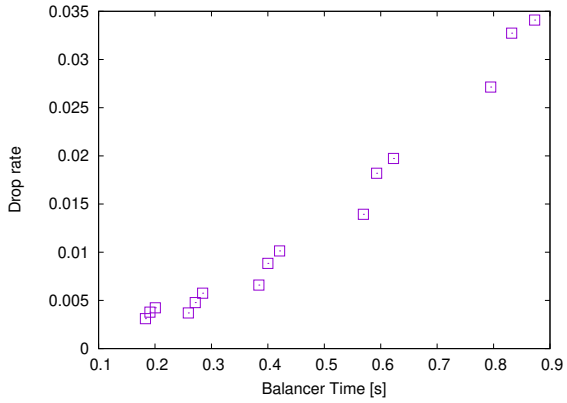
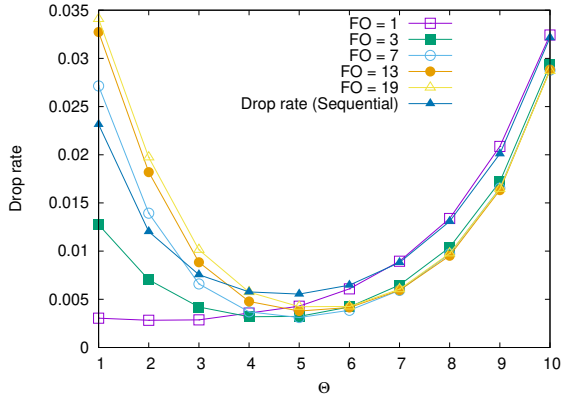
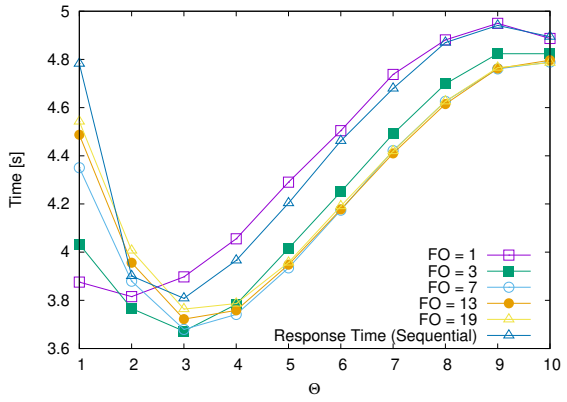


Figure 8: Balancer time *vs.* Drop rate



(a) Drop rate



(b) Response time

Figure 9: Sensitivity to Fan-out FO ($\eta_Q = 10\%$)

Focusing on Fig. 9a we observe that, as FO grows, the curve of the drop rape shifts from a monotone growing shape (when $FO = 1$) to a concave cup-shaped curve. This latter shape, already observed in [3] when studying the sequential forwarding algorithm, occurs when a job is sent to a randomly selected neighbor. 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 and pointed out also in Fig. 8, has a twofold explanation. First, as FO grows, the probe takes longer to complete (because we have to wait for the slowest neighbor and because the query and answer messages can be queued and delayed in the network). 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* [6] may occur so that, when a node reports a load lower than the average, lots of neighbor will select this node as the target for job forwarding, causing a rapid increase in the load of that node.

Focusing on Fig. 9b we observe that the response time of the probe-based algorithm is generally lower compared with the sequential forwarding algorithm with the same threshold value. In particular we observe that, 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, the threshold value that provides the best response time increases from $\Theta = 2$ to $\Theta = 3$, that is towards a case where the number of queries issued is lower.

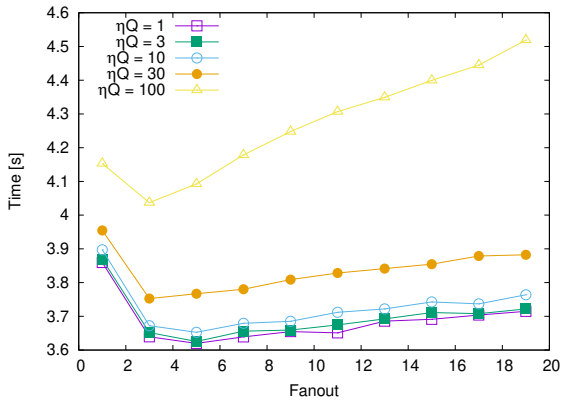


Figure 10: Minimum Response time *vs.* Threshold Θ

Figure 10 summarizes the main findings of the previous analysis on the FO parameter and provides a sensitivity analysis with respect to η_Q, that models the impact of the probe message propagation time with respect to the job forwarding time. To this aim we show the lowest response time over the range of threshold values as a function of the FO and η_Q parameters. We observe that, for every value of η_Q, as FO grows from 1 to 2 we have a reduction in the response time because we are more able to find a less loaded neighbor. When FO grows beyond 2 the minimum response time is increased because the ability to find a less loaded neighbor is counterbalanced by the longer time taken by the probing phase and by the higher risk of experiencing herding effects.

Switching to the analysis of the impact of the η_Q parameter, we observe that, as the probing time grows the overall response time increases due to a higher time required to exchange data across the network. Furthermore, the higher probing time (e.g., when $\eta_Q = 100\%$) increases also the sensitivity of the fan-out value.

5.3 Geographic scenario

We now focus on the simulation results for the geographic scenario previously described.

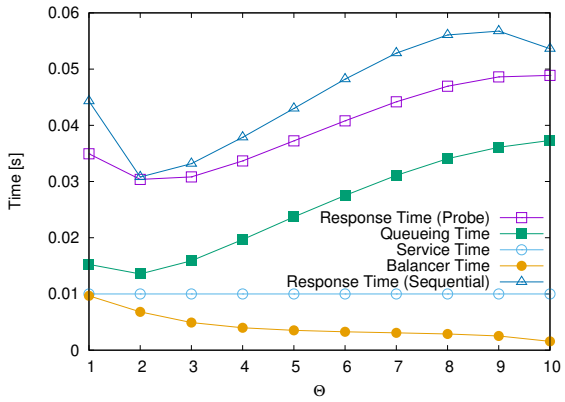
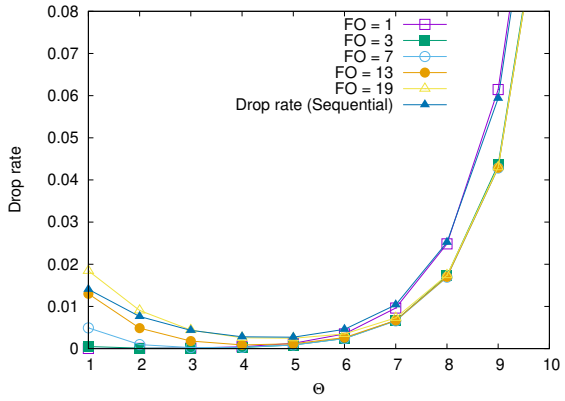


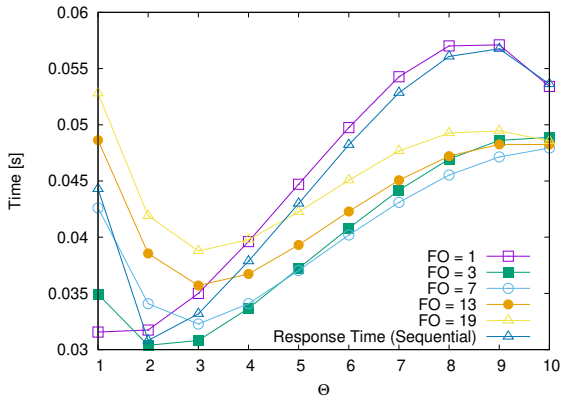
Figure 11: Response time *vs.* Threshold Θ ($FO = 3$, $\eta_Q = 10\%$)

Similarly to Fig. 7 for the uniform mesh scenario, Fig. 11 provides a comparison of the response time achieved by the probe-based and sequential forwarding algorithms when $FO = 3$ and $\eta_Q = 10\%$. Furthermore, a breakdown of the contribution of the response time of the probe-based algorithm is provided. The results confirms the main findings of Fig. 7. The main difference is the better performance of the probe-based solution over the sequential forwarding algorithm, especially for higher values of Θ. Indeed, as this scenario is characterized by an inherent high heterogeneity in the node incoming load, a probe-based approach that can identify nearly idle neighbors provides a major benefit over a purely random neighbor selection.

Figure 12 provides the sensitivity analysis with respect to the query fan-out FO. As a general note we must consider that just a relatively low fraction of the nodes experience overloaded. Hence, only these node will start queries very often. If we consider the theoretical model in Sec. 4, the nodes with a low load experience an effect similar to a reduction of the interval τ. The final result of this behavior is a reduced impact of the herding effect. Observing Fig. 12a we have a clear evidence of this effect because the drop rate curve is quite different from the concave shape observed in Fig. 9a. In particular when Θ is low the drop rate is much lower compared to the case when Θ is close to the maximum. Furthermore, the shape of the drop rate is not highly dependent from the fan-out, unlike what we observed in Fig. 9a. This is an interesting finding that proves how the level of heterogeneity in the infrastructure plays a major role in the benefit achievable through probe-based cooperation. Indeed, probing can take advantage from the presence of an heterogeneous scenario. In our analysis we present also the impact of the fan-out on the response time, but in this case there



(a) Drop rate



(b) Response time

Figure 12: Sensitivity to Fan-out FO ($\eta_Q = 10\%$)

are no unexpected behaviors and the results confirms the main considerations for Fig. 9b. As a further confirmation of the lower impact of the herding effect, we observe also that the minimum response time show a less marked tendency to shift to higher values of Θ as FO grows.

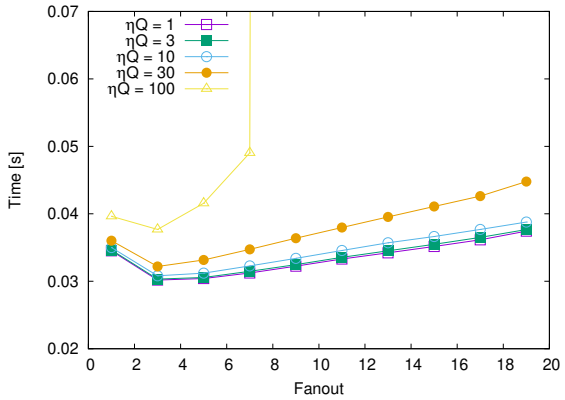


Figure 13: Response time *vs.* Threshold Θ

We conclude our analysis with the impact of the η_Q parameter on the response time shown in Fig. 13. The general shape of the curves confirms the main findings for Fig. 10. However, in this case there is another interesting effect that should be pointed out. Some nodes are issuing a significant amount of queries to cope with the high incoming load. The amount of data exchanged for probing purposes grows with the query message transmission time η_Q and with the fan-out FO. In some cases (e.g., $\eta_Q = 100\%$, $FO > 7$) the data volume can exceed the network capacity, thus resulting in network congestion and in an explosion of the response time.

6 CONCLUSIONS AND FUTURE WORK

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, without relying on centralized workload balancer or reservation mechanisms, with the aim to represent a viable solution for a realistic fog deployment in a smart city scenario possibly characterized by heterogeneous workload distribution. Specifically, 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 expect 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.

The study presented in this paper is part of a broader research line. In future work we plan to investigate different load balancing algorithms based on asynchronous exchange of information about nodes status, and to integrate in the load balancer a predictive mechanism to foresee the nodes status based on machine learning techniques.

REFERENCES

[1] R. Beraldi and H. Alnuweiri. 2019. Exploiting Power-of-Choices for Load Balancing in Fog Computing. In *2019 IEEE International Conference on Fog Computing (ICFC)*. IEEE, Piscataway, New Jersey, US, 80–86.

[2] R. Beraldi, H. Alnuweiri, and A. Mtibaa. 2018. A Power-of-Two Choices Based Algorithm for fog Computing. Early Access. *IEEE Transactions on Cloud Computing* (2018), 1–1. https://doi.org/10.1109/TCC.2018.2828809

[3] 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.

[4] 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.

[5] S. Chen, T. Zhang, and W. Shi. 2017. Fog Computing. *IEEE Internet Computing* 21, 2 (2017), 4–6.

[6] M. Dahlin. 2000. Interpreting Stale Load Information. *IEEE Transactions on Parallel and Distributed Systems* 11, 10 (oct 2000), 1033–1047.

[7] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang. 2016. Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption. *IEEE Internet of Things Journal* 3, 6 (Dec 2016), 1171–1181.

[8] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis. 2017. A Cooperative Fog Approach for Effective Workload Balancing. *IEEE Cloud Computing* 4, 2 (March 2017), 36–45. https://doi.org/10.1109/MCC.2017.25

[9] Evgeny Khorov, Andrey Lyakhov, Alexander Krotov, and Andrey Guschin. 2015. A survey on IEEE 802.11 ah: An enabling networking technology for smart cities. *Computer Communications* 58 (2015), 53–69.

[10] Eli Upfal Michael Mitzenmacher. 2005. *Probability and computing: randomized algorithms and probabilistic analysis*. CAMBRIDGE UNIVERSITY PRESS, The Edinburgh Building, Cambridge CB2 2RU, UK.

[11] M. Mitzenmacher. 2000. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems* 11, 1 (2000), 6–20.

[12] OpenFog Consortium Architecture Working Group. 2017. *OpenFog Reference Architecture for Fog Computing*. Technical Report OPFRA001.020817. OpenFog Consortium Architecture Working Group.

[13] E. C. Pinto Neto, G. Callou, and F. Aires. 2017. An algorithm to optimise the load distribution of fog environments. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Piscataway, New Jersey, US, 1292–1297.

[14] 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.

[15] Mahadev Satyanarayanan, Wei Gao, and Brandon Lucia. 2019. The Computing Landscape of the 21st Century. In *Proc. of the 20th International Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, USA) *(HotMobile '19)*. 45–50.

[16] Y. Song, S. S. Yau, R. Yu, X. Zhang, and G. Xue. 2017. An Approach to QoS-based Task Distribution in Edge Computing Networks for IoT Applications. In *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE, Piscataway, New Jersey, US, 32–39.

[17] 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.

[18] 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.