PAFFI: Performance Analysis Framework for Fog Infrastructures in realistic scenarios

Claudia Canali*, Riccardo Lancellotti*
*Department of Engineering ”Enzo Ferrari”, University of Modena and Reggio Emilia,
via Vivarelli 10, Modena, Italy
Email:{claudia.canali, riccardo.lancellotti}@unimore.it

***Abstract*—The growing popularity of applications involving the process of a huge amount of data and requiring high scalability and low latency represents the main driver for the success of the fog computing paradigm. A set of fog nodes close to the network edge and hosting functions such as data aggregation, filtering or latency sensitive applications can avoid the risk of high latency due to geographic data transfer and network links congestion that hinder the viability of the traditional cloud computing paradigm for a class of applications including support for smart cities services or autonomous driving. However, the design of fog infrastructures requires novel techniques for system modeling and performance evaluation able to capture a realistic scenario starting from the geographic location of the infrastructure elements. In this paper we propose PAFFI, a framework for the performance analysis of fog infrastructures in realistic scenarios. We describe the main features of the framework and its capability to automatically generate realistic fog topologies, with an optimized mapping between sensors, fog nodes and cloud data centers, whose performance can be evaluated by means of simulation.**

***Index Terms*—Fog computing, Framework, Optimization problems, Simulation**

I. Introduction

The fog computing paradigm has received an increasing attention as a solution to support applications characterized by a huge amount of data to process and/or with specific requirements related to low and predicable latency [1], [2]. Such characteristics are typical of increasingly popular applications where IoT sensors, geographically distributed over a wide area, continuously collect data that need to be processed, filtered and aggregated to extract useful information for specific application purposes. Meaningful examples of this kind of applications are related to smart cities scenarios, such as traffic monitoring, support for autonomous driving, smart grids, environmental monitoring and management, public safety, etc. These applications are traditionally handled through cloud computing systems, that are represented in the left part of Fig. 1: in these architectures, the data collected by the sensors are directly sent from the *sensor layer* to the *cloud layer* where the data center(s) are located to be processed and stored. However, this approach may lead to high and unpredictable latency due to the geographic data transfer; furthermore, we are exposed to the risk of network congestion as the entire amount of data is sent to the cloud servers to be processed, even in the cases where data could be pre-processed (aggregated and filtered) to store in the network core only a reduced set of meaningful data. The emerging paradigm of fog computing represents a solution addressing the cloud computing limitations through the use of an intermediate layer of *fog nodes* that are interposed between the cloud data center(s) and the sources of the data, as shown in the right part of Fig. 1. Basically, the fog paradigm extends the cloud computing by moving (part of) the computation at the *network edge*, where the fog nodes are located. This decreases the latency experienced by the applications and reduces the amount of information transferred to the network core that is limited to the output of the pre-processing tasks carried out at the fog layer [1], [2].

The use of a fog computing infrastructure has been only quite recently addressed in literature. Most of the existing studies focus on the part of the infrastructure included between the fog layer and the cloud data center(s): for example, the studies in [3], [4] address the issue of optimizing the allocation of the processing tasks sent by the fog nodes towards the cloud servers. On the other hand, the mapping of data sources over the fog nodes is usually not taken into account, assuming a single hop wireless links between sensors and fog nodes [3] or a fixed mapping that typically connects each sensor to the closest fog node [5]. This first level of the fog infrastructure is considered only in few studies in literature, such as [6], where a formal model is proposed to optimize the mapping of the workload coming from the sensors over the fog nodes.

In general, the exploitation of the fog computing paradigm raises many new challenges: in [1], [2], the main open issues and research directions about fog infrastructures supporting IoT services and smart cities are identified and discussed. A further open problem is the a lack of instruments supporting the design and the performance analysis of distributed fog computing infrastructures in realistic scenarios.

The main contribution of this paper is the proposal of a framework, namely PAFFI (Performance Analysis Framework for Fog Infrastructures), specifically designed to evaluate the performance of a fog computing system considering different alternative scenario parameters, including the mapping of the data sources over the available fog nodes, by means of simulation. The wide range of possible options in terms of resource allocation and management within such complex distributed systems, indeed, makes simulation the ideal tool to investigate

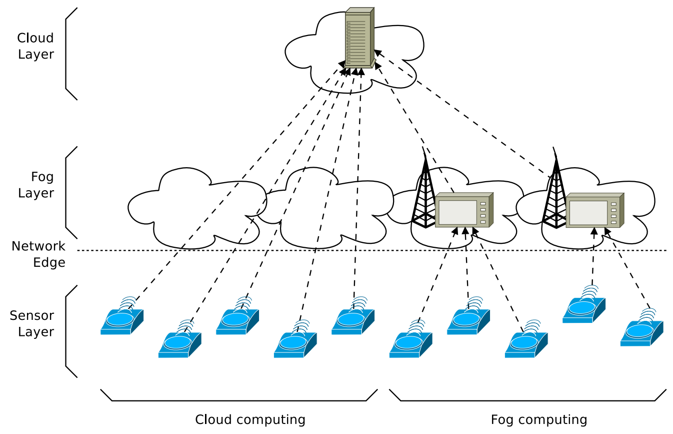


Fig. 1: Cloud and Fog Infrastructures

alternative strategies and settings. In order to provide high flexibility and adaptability to different scenarios, the framework has been designed according to a modular approach based on building blocks that can be easily modified or re-arranged to integrate new functionalities and options. Specifically, the proposed framework takes as input a list of points of interest coming from a real environment (e.g., addresses or street names) that will be used as the input to define the location of the sensors, the fog nodes and the cloud data center(s). These entries are then geo-referenced by exploiting online available services; the network delay between the elements of the fog infrastructure is considered proportional to their geographic distance. Then, the actual topology is generated considering different options for mapping sensors over fog nodes and fog nodes over cloud data center(s), including not only a naive mapping based on the closest distance but also an optimized mapping based on the solution of a formal model to minimize the overall latency and processing time. The final step concerns the performance evaluation of the overall fog infrastructure: starting from the identified topology, the input scenario for a network simulator is automatically generated considering a model based on the queuing theory; finally, the output of the simulation is analyzed to extract the main performance metrics. To the best of our knowledge, PAFFI is the first general purpose framework for fog analysis that introduces the support for realistic scenario obtained through geo-referencing.

In this paper, the proposed framework is evaluated in a specific case study based on the design of a prototype implementation of a smart city sensing application. The obtained results show how the PAFFI framework enables an easy and fast comparison of different alternatives to identify the best topology of the fog infrastructure; moreover, the experiments demonstrate how the optimized mapping brings significant improvements in the performance, avoiding the risk of overload conditions over the fog nodes.

The rest of the paper is structured as follows. Section II presents an overview of the fog infrastructure and of the performance problems. Section III describes the core part of the proposed framework, that is the mechanism for the generation of the fog scenarios to evaluate. Section IV presents the case study. Finally, Section V discusses some related work and Section VI presents some concluding remarks and discusses some future research directions.

II. Problem definition

In this paper we consider a fog infrastructure, such as the one represented in the right part of Fig. 1, consisting of three layers: a *sensor layer* including wireless sensors that collect and produce data, a *fog layer* responsible for a preliminary processing of data from the sensors and a *cloud layer* which is the final destination of the data. We assume that the considered fog infrastructure supports smart cities applications, such as traffic monitoring or environmental sensing. In this scenario, the sensors collect the information about the city status of interest (e.g., traffic intensity, air quality [7], U.V. rays intensity). The collected data are sent to the intermediate fog layer for pre-processing tasks (e.g., filtering and aggregation) with the twofold aim of reducing the latency perceived by the smart cities application and decreasing the network traffic towards the cloud servers located at the network core, where the pre-processed data are finally sent to be stored and/or further processed to provide value-added services such as traffic or pollution forecast [7].

The problem of performance evaluation in such a complex and geographically distributed infrastructure concerns several aspects, including the management of data flows from sensors to fog nodes and from fog nodes to cloud data center(s). It is worth to note that in this paper we do not take into account the inner dynamics of the cloud data centers because several solutions have been already proposed in literature for that level [8], [9], focusing instead on the management of the data flows at the lower levels.

To formally describe our problem, we assume a stationary scenario with a set of $\mathcal{S}$ similar sensors distributed over a geographical area. Furthermore, we assume that sensors are producing data at a steady rate, with a frequency that we denote as λ_i for the generic sensor i. We anticipate that out case study (and the framework code supporting this feature) will focus on all sensors producing data with the same rate, but it is straightforward to enrich the framework to support heterogeneous data rates for the sensors, without the need to update the underlying model. The fog layer consists of a set of $\mathcal{F}$ nodes receiving data from the sensors and performing operations on them. These operations typically include pre-processing of the data, such as filtering and/or aggregation, or may include some form of analysis to identify anomalies or problems as fast as possible. We denote the processing rate of the generic fog node j as μ_j (also for the fog nodes we assume an homogeneous nature, but we can extend the framework to handle also heterogeneous fog nodes scenarios). The refined data samples from the fog nodes are then sent to the cloud data center $k \in \mathcal{C}$, where additional analysis can be carried out and the information is stored.

The geographically distributed nature of the fog infrastructure is modeled using two matrices of distances: $\delta_{i,j}$ to describe the sensors-to-fog network delays, and $\delta_{j,k}$ for the fog-to-cloud delays.

For a complete summary of the symbols used in the problem formalization, refer to Tab. I.

TABLE I: Notation.

Symbol	Meaning/Role
Model parameters	
$\mathcal{S}$	Set of sensors
$\mathcal{F}$	Set of fog nodes
$\mathcal{C}$	Set of cloud data centers
λ_i	Outgoing data rate from sensor i
λ_j	Incoming data rate at fog node j
$1/\mu_j$	Processing time at fog node j
$\delta_{i,j}$	Communication latency between sensor i and fog node j
$\delta_{j,k}$	Communication latency between fog j and cloud k
Model variables	
i	Index of a sensor
j	Index of a fog node
k	Index of a cloud data center
Data flows description	
$x_{i,j}$	Data flow from sensor i to fog node j
$y_{j,k}$	Data flow from fog node j to cloud data center k
Scenario description	
δ	Average network delay
$\delta\mu$	Network delay to processing time ratio
ρ	Infrastructure load

To define the data flows, we introduce two matrices of boolean variables. First, matrix $X = \{x_{i,j}\}, i \in \mathcal{S}, j \in \mathcal{F}$, describes the sensors-to-fog mapping in such a way that $x_{i,j} = 1$ if sensor i sends data to fog node j, while $x_{i,j} = 0$ if this data exchange does not occur. To support some stateful pre-processing (stateful pre-processing includes also trivial aggregation such as a moving window average), we assume that all the data of a sensor should be sent to the same fog node. Hence, for each i, there is only one value of j such that $x_{i,j} = 1$. Second, matrix $Y = \{y_{j,k}\}, j \in \mathcal{F}, k \in \mathcal{C}$, captures the fog-to-cloud mapping with $y_{j,k} = 1$ if fog node j sends data to cloud data center k, while $y_{j,k} = 0$ if this data exchange does not occur. As for the sensor-to-fog mapping, each fog nodes sends all its data to just one cloud data center.

Given a description of the data flows, we consider useful to introduce a symbol to describe the amount of data arriving at a fog node λ_j, defined as:

$$\lambda_j = \sum_{i \in \mathcal{S}} x_{i,j} \cdot \lambda_i \tag{1}$$

To describe the model, the main parameters of interest are the sensors data rate λ_i, the processing rate μ_j and the network delays $\delta_{i,j}$ and $\delta_{j,k}$. Applying the Queuing Theory to this problem formulation, similarly to the modeling in [6], we can define the expected performance of a given sensor-to-fog and fog-to-cloud mappings as follows:

$$T_{netsf} = \frac{1}{\sum_{i \in \mathcal{S}} \lambda_i} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} \lambda_i x_{i,j} \delta_{i,j} \tag{2}$$

$$T_{netfc} = \frac{1}{\sum_{j \in \mathcal{F}} \lambda_j} \sum_{j \in \mathcal{F}} \sum_{k \in \mathcal{C}} \lambda_j y_{j,k} \delta_{j,k} \tag{3}$$

$$T_{proc} = \frac{1}{|\mathcal{F}|} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} x_{i,j} \cdot \frac{1}{\mu_j - \lambda_j} \tag{4}$$

Specifically, T_{netsf} is the average network delay (weighted by the data flow intensity in the sensor-to-fog mapping) experienced by the jobs passing from the sensors to the fog nodes. Similarly, T_{netfc} is the network delay from the fog nodes to the cloud data centers. Finally, T_{proc} is the response time (including processing and queuing) at the fog node. This definition is obtained considering a fog node as a M/G/1 queuing system (that is with a Poisson arrival stochastic process with average inter-arrival time $1/\lambda_j$) and a processing time that is typically described with a Gaussian probability with a mean $1/\mu_j$ and a standard deviation σ_j.

The parameters λ_i, μ_j, $\delta_{i,j}$, and $\delta_{j,k}$ are used to estimate the expected response time of a fog infrastructure. However, when defining experiments and designing a fog scenario, we consider more useful to rely on simplified parameters such as the average network delay δ, the ratio between the network delay and the processing time $\delta\mu$ or the load on the infrastructure ρ. Moreover, each of these parameters may be a useful driver for a sensitivity analysis. The parameters are defined as follows, and from them it is straightforward to derive the other model parameters used in the above equations.

$$\delta = \frac{\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} \delta_{i,j} + \sum_{j \in \mathcal{F}} \sum_{k \in \mathcal{C}} \delta_{j,k}}{|\mathcal{S}| \cdot |\mathcal{F}| + |\mathcal{F}| \cdot |\mathcal{C}|} \tag{5}$$

$$\delta\mu = \delta \cdot \frac{\sum_{j \in \mathcal{F}} \mu_j}{|\mathcal{F}|} \tag{6}$$

$$\rho = \frac{\sum_{i \in \mathcal{S}} \lambda_i}{\sum_{j \in \mathcal{F}} \mu_j} \tag{7}$$

III. Framework for fog scenarios generation

In this section we describe the proposed PAFFI framework for the generation of fog scenarios starting from a realistic setting for smart cities applications.

Fig. 2 shows the overview of the framework according to the BPMN specification. The framework consists of three main building blocks, namely *Geo-referencing*, *Scenario generation* and *Performance evaluation*, implementing the main steps of the process that leads, starting from a simple list of points of interest (POIs), to the generation of a fog infrastructure for a smart city application, with the possibility to easily explore a wide range of scenario setups and parameters, and finally to its evaluation based on performance metrics such as response and queuing times. Each of the main steps exploits external services to carry out the required functionalities.

The reference language of the entire framework is Python. It is worth to note that the modular approach followed in the overall design of our proposal as well as the choice to rely on well known open source technologies for the framework implementation and on external online services allows the developers to easily extend the proposed framework to modify

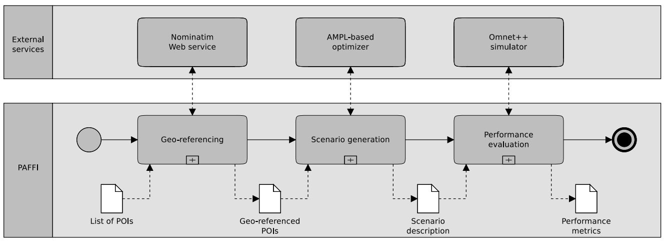


Fig. 2: BPMN framework overview

functionalities and integrate new features. Every module of the PAFFI framework, indeed, can be easily modified or substituted with the only requirement to follow the input/output formats established for the data exchange between modules. Furthermore, the modular structure of the framework enables and facilitates what-if-analysis about the performance of the fog infrastructure. The framework will be made available by the authors on request.

The details of each step, shown in Fig. 3, are described in the rest of this section.

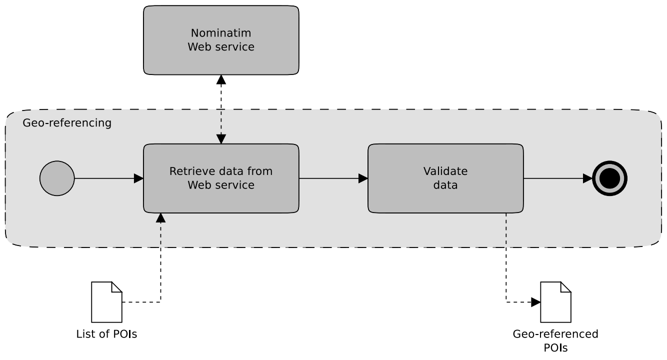
A. Geo-referencing

The first building block of the proposed framework is responsible for the geo-referencing of the main elements of the fog infrastructure, that are sensors, fog nodes and cloud data centers. The process is schematized in Fig. 3a. The input for the geo-referencing module is represented by a lists of POIs (one entry per row) expressed as references to the real environment where the smart city application will be developed: specific addresses or street names (in this case the POI is considered in the geometric center of the street) where the elements of the fog infrastructure will be actually located. The process is iterated three times, considering one separated list for each category of elements (sensors, fog nodes and cloud data centers). Then, the geo-referencing module parses the POIs list and retrieves the coordinates of the indicated points from the external Web service OpenStreetMap Nominatim[1]. The data are finally validated, removing the entries whose geographic reference was not available through the online service, and producing as output the geo-referenced lists with geographic coordinates associated to each entry.

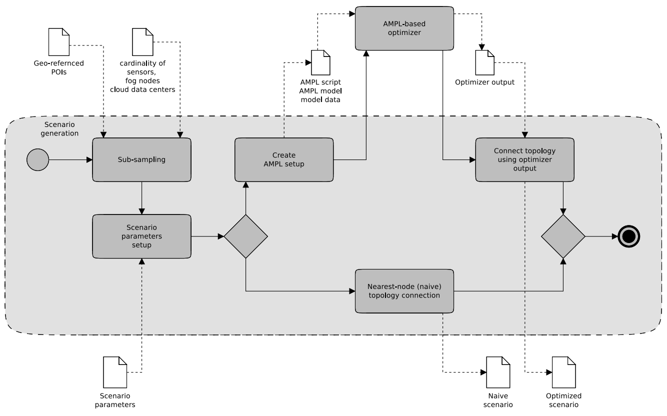
B. Scenario generation

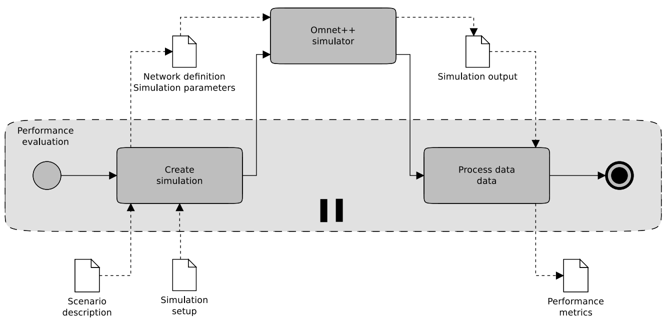
The final aim of the second building block is the generation of the fog scenario for the smart city application. As shown in Fig. 3b, the input for this process consists of the lists of geo-referenced POIs along with the number of sensors, fog nodes and cloud data centers that will actually compose the fog infrastructure: for each infrastructure element, a number of locations equal to the corresponding input is randomly extracted from the corresponding geo-referenced list. Once the final locations of all the elements of the fog infrastructure are defined, the sensor-to fog and the fog-to-cloud distances are computed using the haversine formula. The distances are

[1]https://wiki.openstreetmap.org/wiki/Nominatim



(a) Geo-referencing


AMPL-based optimizer
Geo-refenced POIs
cardinality of sensors, fog nodes cloud data centers
AMPL script AMPL model model data
Optimizer output
Scenario generation
Sub-sampling
Create AMPL setup
Connect topology using optimizer output
Scenario parameters setup
Nearest-node (naive) topology connection
Scenario parameters
Naive scenario
Optimized scenario
(b) Scenario generation



(c) Performance evaluation

Fig. 3: Details of PAFFI Building Blocks

then scaled according to the vale of δ to obtain the network delays $\delta_{i,j}$ and $\delta_{j,k}$ according to Eq. (5). Further inputs are the values of the parameters defining the scenario in terms of average load on the infrastructure (ρ) and ratio between network delay and processing time ($\delta\mu$). Given their values, it is straightforward to derive the parameters λ and μ from the Equations (7) and (6): as anticipated in the previous section, we assume a homogeneous nature for the sensors, producing data at the same rate, and for the fog nodes, characterized by the same processing rate.

At this point, the connected topology of the fog infrastructure has to be defined in order to determine how the data flow from sensors to fog nodes and then from fog nodes to cloud data centers. To define the sensors-to-fog and the fog-to-cloud mappings (expressed by X and Y matrices in the model defined in Section II), we have two options. First, a naive mapping selecting the closest node based on the geographical distance: in this case, the sensors will send data to the nearest fog node, which is a typical approach in literature [3], [5]. Second, an optimized mapping that, based on the scenario parameters, minimizes the average network delay and response time experienced by the elements of the fog infrastructure: basically, this mapping minimizes the sum of the contributions to the infrastructure performance T_{netsf}, T_{netfc} and T_{proc} defined in Equations (2), (3), and (4), respectively.

The naive topology connection can be easily obtained by computing the geographic distances between the element of the infrastructure starting from the geo-referenced lists of POIs to finally produce the naive fog scenario description as the output of the second building block of the PAFFI framework. On the other hand, the generation of the optimized topology is a more complex task requiring the generation and the resolution of the corresponding optimization problem. To this aim, the proposed framework is able to automatically generate the optimization problem based on the scenario parameters and implemented with the AMPL modeling language [10], creating the AMPL files `.mod`, `.dat` and `.run` required by solvers. Then, the framework invokes an external AMPL-based optimizer (such as KNITRO[2]) that solves the problem and generates the optimized sensors-to-fog and fog-to-cloud mappings. The complete formulation of the optimization problem is similar to the one proposed in [6]. Finally, the framework parses the solver output and produces the optimized fog scenario description. In both cases (naive and optimized), the scenario description is a file in JSON format containing: the lists of sensors, fog nodes and cloud data centers; the outgoing data rates λ_i from the sensors; the matrices of the connections $X = \{x_{i,j}\}, i \in \mathcal{S}, j \in \mathcal{F}$ and $Y = \{y_{j,k}\}, j \in \mathcal{F}, k \in \mathcal{C}$; the network delays $\delta_{i,j}$ and $\delta_{j,k}$; the processing times $1/\mu_j$ at the fog nodes.

C. Performance evaluation

The last framework building block is responsible to carry out the performance evaluation of the overall fog system by means of simulation. To this aim, the building block takes as input a JSON description of the fog scenario along with the simulation setup, as shown in Fig. 3c; starting from this information, the input scenario for a network simulator is automatically generated considering a model based on the M/G/1 queuing theory. To run the simulation, we make use of the discrete event network simulator OMNeT++ [11]; hence, two files are created as input for the simulator: the `.ned` file containing the network description and the `.ini` file containing all the other simulation parameters. After running the simulation, the output of the simulator is processed and analyzed to extract the main performance indicators for the fog scenario: the final output of the performance analysis includes indicators extracted both at the level of the fog nodes and of the cloud data centers. For each fog node, we measure the node utilization, the queue length and the time spent by the jobs in the node waiting queue; at each cloud data center, we extract the average response time, queuing time and processing time experienced by all the incoming jobs through their path from the sensors to the cloud.

[2]https://www.artelys.com/solvers/knitro/

IV. Experimental results

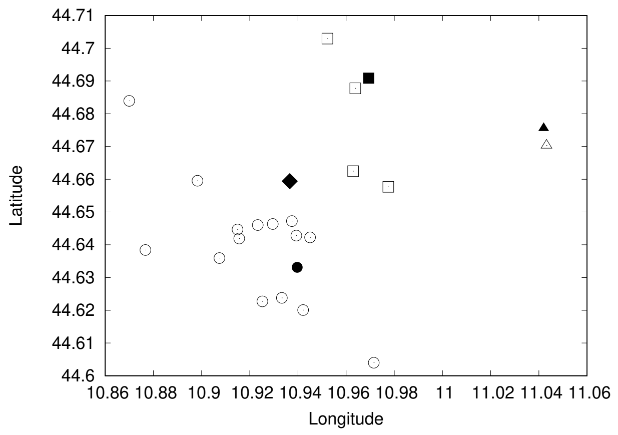
A. Case study description

The considered case study concerns the design of a prototype implementation of a smart city application, where several sensors collect information on the environment, such as air quality [7], traffic (i.e., number of vehicles passing on the streets) or other metrics (such as U.V. rays intensity) to monitor in real time several metrics related to the quality of living in a city. The prototype aims to demonstrate the viability of fog computing for this kind of application. Due to the prototype nature of the infrastructure, we consider the number of sensors and fog nodes to be limited. In particular, in this test case, we consider 20 sensors and 3 fog nodes. Furthermore, due to the small geographic area involved, we assume that just one cloud data center is enough to collect all the data and to support any smart city application. The locations hosting the elements of the fog scenarios are randomly selected from a list of suitable candidates: the locations for the sensors are selected from the list of the streets of the city, while the fog nodes locations are selected from a list of public buildings belonging to the municipality; the cloud data center is located in the site of the municipality data center.

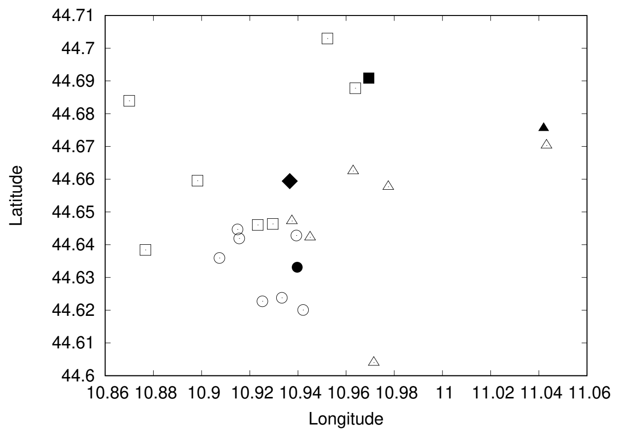
Other significant scenario parameters are the outgoing data rate of the sensors, the processing capacity of the fog nodes and the network delays. These parameters can be derived by the scenario parameters ρ, $\delta\mu$ and δ. Specifically, in our experiments we consider $\rho = 0.5$, which means that the fog infrastructure is only half utilized on average, leaving space to handle traffic surges, increasing complexity of the deployed services, or to some degree of unbalance in the workload. For the impact of network delays, we consider an average delay in the geographic links in the order of $\delta = 10$ ms (the order of magnitude is obtained from preliminary experimental tests) and $\delta\mu = 1$, meaning that the service time is comparable with the network delay (preliminary tests suggested that this scenario is the most interesting as both load balancing and network topology optimization play a role in the resulting performance). From these parameters we infer $\lambda_i = 7.5$ jobs/s and $\mu_j = 100$ jobs/s. It is worth to note that the proposed framework supports an easy exploration of a large space of scenario setups. For space reasons we present just one analysis, that we consider most significant from a smart city point of view, concerning the impact of the fog infrastructure topology on the performance.

B. Comparison between Naive and Optimized Mapping

The main comparison carried out through our experiments is about the impact of the topology mapping on the fog infrastructure performance: it is worth to note that, as we have only one cloud data center in our case study, the problem is limited to the sensors-to-fog mapping, for which we compare two main alternatives: the previously-described naive mapping, where each sensors sends data to the nearest fog node, and the alternative mapping based on the solution of an optimization problem as described in Section III-B.



(a) Naive mapping



(b) Optimized mapping

Fig. 4: Naive and Optimized mapping

A visual representation of the two mappings is shown in Fig. 4: Fig. 4a shows the naive mapping, while Fig. 4b presents the optimized mapping. Both figures present the position of data center (filled diamond) and of the three fog nodes (filled square, circle and triangle). Furthermore, we show the location of the sensors, whose (non-filled) shape reveals their mapping over the fog nodes: the shape of the sensor, indeed, is the same of the fog node receiving its data flow. If we focus on the naive mapping in Fig. 4a, we observe that the triangle fog node receives data from just one sensor, while the circle fog node is at risk of overload, due to the high number of sensors it is connected to. The optimized mapping in Fig. 4b corrects this unbalancing and provides a better load sharing among the fog nodes.

C. Performance analysis

The performance of the two fog scenarios, characterized by a different sensors-to-fog mapping, are compared using the OMNeT++ simulator [11]. We rely on the standard modules provided by the simulator to capture the details of the scenario. As the considered problem representation is based on an M/G/1 queuing theory formalization, our simulation implements a Poisson arrival process (with an average inter-arrival time $1/\lambda_i = 133$ ms) from the sensors and a Gaussian processing time distribution characterized by an average time $1/\mu_j = 10$ ms and a standard deviation $\sigma_j = 2.2$ ms. The standard deviation of the processing time is obtained from the prototype implementation of a sensing application that collects frames from a video and extracts useful information such as the presence of vehicles or pedestrians. The simulation is limited to just 5 minutes because some dynamics that characterize our scenarios (such as the occurrence of overload) make even the transient periods meaningful, without the need to wait for a steady state. Each simulation is run 10 times and the results are averaged to guarantee that the results are statistically significant.

TABLE II: Performance comparison

Parameter	Naive mapping	Optimized mapping
	Fog node statistics	
Utilization	0.30, ≈ 1, 0.075	0.54, 0.53, 0.45
Queue length	0.07, ≥ 1987, 0.0031	0.33, 0.31, 0.19
Queuing time [ms]	2.2, ≥ 17650, 0.41	6.0, 5.9, 4.1
	Cloud statistics	
Response time [ms]	≥ 12807	30.8
Queuing time [ms]	≥ 12786	5.4
Processing time [ms]	10	10

From the simulation runs we extract a set of performance indicators, summarized in Tab. II, both at the level of the cloud data center and of the fog nodes. To better understand the behavior of the system under the two different mappings, we start analyzing the performance at the level of fog nodes: in the first part of the table, we report the values (one for each fog node) of three considered metrics, that are node utilization, size of the queue and time spent by the jobs in the waiting queue. Starting from the utilization of the fog nodes (that is the fraction of time spent by the node processing jobs) we observe that the naive mapping determines a clear unbalancing in the load distribution at the second fog node (the one represented as a circle in Fig. 4a) showing signs of overload (as the time evolution of the node performance metrics suggests that the node is not in a steady state, the reported values for this node are approximations or lower bounds), with a utilization basically equal to one. The overload on this node is further testified by the very high queue length and by the corresponding time spent by the jobs in the waiting queue. The time evolution of the fog node queue lengths for two fog nodes is shown in Fig. 5; as the values spans over multiple orders of magnitude, we use a logarithmic scale for the y axis. The node fog$_0$, marked with a filled square in Fig. 4a, presents a queue length that oscillates between 0 and 7 jobs, without clear trends; on the other hand, the node fog$_1$, that is the node marked as a filled circle in Fig. 4a and presenting signs of overload, shows a queue evolution that grows over time (the growth is linear even if, due to the logarithmic scale, the curve is not shaped as a straight line), consistently with an overload condition.

Considering the optimized mapping, instead, we observe a fair level of load sharing resulting in a similar utilization of every node, and with small queue lengths (on average less than 1). As a consequence, the average queuing time is in the order of 4-6 ms, that is comparable with the service time (that is 10 ms).

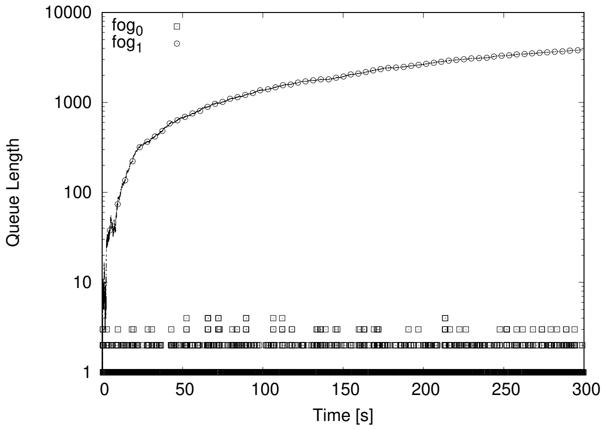
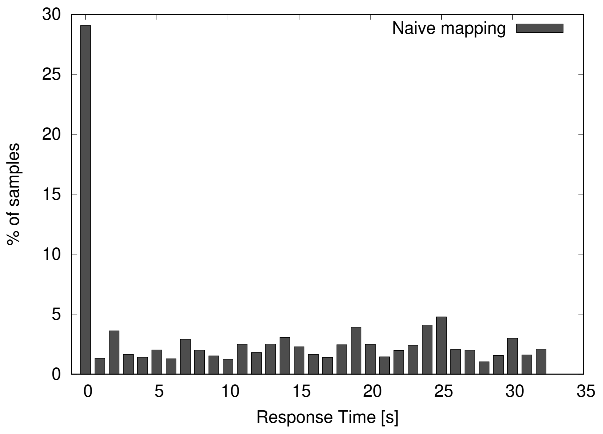


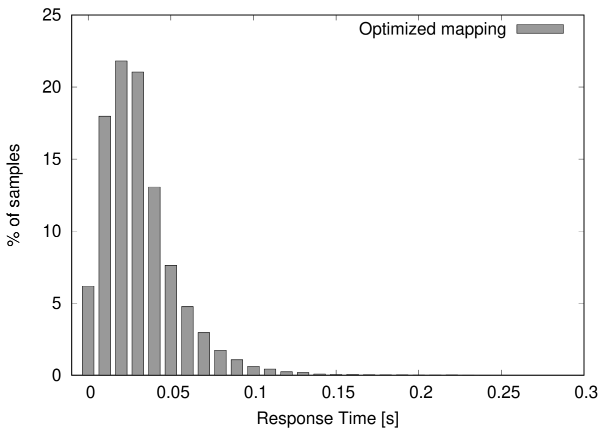
Fig. 5: Queue lenght analysis

Switching to the data collected at the cloud data center, the average response time experienced by the incoming jobs (intended as the time spent between the initial transmission by the sensors and the arrival at the cloud) is the most straightforward performance indicator. A comparison of this metric clearly shows how the optimized mapping outperforms the naive mapping. Indeed, the response time for the naive mapping explodes (the value grows constantly as the simulation time increases). Since the service time for every job is the same for every simulated scenario (as shown in Table II) and the naive mapping explicitly aims at reducing the network delay, the different performance is due to the queuing time in the fog nodes, as shown by our experiments. This result is consistent with the overload in one fog node previously pointed out when discussing the simulation results at the level of fog nodes.

The effect of the overload on a fog node can be observed also if we consider the probability distribution of response time as shown in Fig. 6. In Fig. 6a we observe the presence of two separate sets of values. The first column of the histogram, with a large number of requests characterized by a response time of less than 1 s, is related to the jobs processed by the non-overloaded nodes. On the other hand, the jobs processed by the overloaded node (accounting for a large fraction of the total requests, due to the number of sensors sending data to



(a) Naive mapping



(b) Optimized mapping

Fig. 6: Response time

that fog node) experience a response time evenly distributed over a wide range of values, with a response time that grows over time with the size of the queue, shown in the remaining columns of the histogram in Fig. 6a. When no overload occurs, such as for the optimized mapping shown in Fig. 6b, the overall response time is orders of magnitude smaller, and the probability distribution follows a curve that depends on the Gaussian distribution of the service time, on the network delay and on the queuing time.

V. Related work

The benefits of fog computing for applications requiring to manage a large amount of (possibly latency-sensitive) data compared to a traditional cloud architecture has been widely pointed out in literature. For example, Wen *et al.* [1] and Yi *et al.* [2] both discuss the main benefits, the core challenges and the issues in fog system, with the latter providing special attention on the scalability of the service orchestration in such scenario.

When considering performance and scalability in fog computing, it is common to consider network-related delays as the main metric, as in [4], even if more complex metrics that consider both network delay and processing time [6] or that take into account power consumption [3] have been proposed. While our approach is more tailored to the model used in [6], we believe that the PAFFI framework is very flexible and can extended to support also other performance metrics and models.

Also the focus on the layers of a fog infrastructure is highly differentiated in literature. Some papers focus on the fog-to-cloud interaction, assuming that the mapping of sensors over the fog nodes is based on the limited range of wifi or bluetooth connections [3] or fixed due to design choices [5]. In these cases workload unbalancing is handled through job redirection that is decided on the basis on some optimization problem [3], [4] or using some adaptive algorithm that runs on each fog node [12]. Other studies explicitly focus on the mapping of sensors over the fog nodes [6], assuming either a wide-range wireless connection (such as, LoRa) or some technique for network traffic engineering in the fog infrastructure.

Furthermore, concerning the techniques used for performance evaluation, multiple approaches are available. Some papers rely on purely theoretical models based on MILP such as [4], [6], [13], with the latter paper focusing on a more general problem related to data stream processing characterized by a model similar to the one we consider in fog computing. Other papers rely on simulators, both ad-hoc developed simulators (for example based on MATLAB) [3], [12], and general purpose simulators [14]; for a discussion on the simulation of fog systems, the reader may refer to [15]. Finally, some papers base their evaluation on small scale prototypes [5]. Our framework fits perfectly in this heterogeneity as it supports both theoretical models and simulation, integrating an interface to interact with the Omnet++ [11] simulation engine.

As a closing remark, limited effort has been devoted to the problem of creating realistic scenarios for a geographically distributed network. While some tools have been used to model fog computing platforms that manage geographical data [14], they typically lack a general-purpose support to create a realistic fog infrastructure description. The most common approach is to simply consider uniformly-generated random values for the network latency [4]. Our framework provides a major contribution with respect to this issue introducing the support to design realistic network topologies starting from lists of POIs.

VI. Conclusions and future work

In this paper we presented a flexible and modular framework supporting the design and the performance evaluation of a fog infrastructure for smart cities applications where data flows generated by IoT sensors are pre-processed by fog nodes before going to cloud data centers for final storage and processing. Starting from simple lists of POIs, expressed as addresses or street names, and from few parameters defining the expected workload and the processing capabilities of the nodes, the proposed framework is able to generate realistic fog scenarios with optimized sensors-to-fog and fog-to-cloud mappings. Evaluated in a specific case study related to the design of a smart city sensing application, the results show how the PAFFI framework provides a useful evaluation of parameters and alternatives for the identification of the best fog infrastructure. The proposed framework represents a work in progress that will be extended in future works. In particular, we plan to add new modules to include fog-to-fog cooperation strategies to improve the load balancing in processing the data flows coming from the sensors, supporting the cooperation mechanism both at the level of topology generation and of simulation.

References

[1] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, Mar 2017.

[2] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. ACM, 2015, pp. 37–42.

[3] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec 2016.

[4] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *2017 IEEE International Conference on Edge Computing (EDGE)*, June 2017, pp. 17–24.

[5] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proceedings of the ASE BigData & SocialInformatics 2015*, ser. ASE BD&SI '15. New York, NY, USA: ACM, 2015, pp. 28:1–28:6.

[6] C. Canali and R. Lancellotti, "A Fog Computing Service Placement for Smart Cities based on Genetic Algorithms," in *Proc. of International Conference on Cloud Computing and Services Science (CLOSER 2019)*, Heraklion, Greece, May 2019.

[7] A. Bigi, G. Veratti, S. Fabbi, O. Ziven, L. Po, and G. Ghermandi, "Forecast of the impact by local emissions at an urban micro scale by the combination of lagrangian modelling and low cost sensing technology: the TRAFAIR project," in *Proc. of 19th International conference on Harmionisation within Atmospheric Dispersion Modelling for Regulatory Purposes*, Bruges, Belgium, June 2019.

[8] C. Canali and R. Lancellotti, "Scalable and automatic virtual machines placement based on behavioral similarities," *Computing*, vol. 99, no. 6, pp. 575–595, June 2017.

[9] L. Wei, C. H. Foh, B. He, and J. Cai, "Towards Efficient Resource Allocation for Heterogeneous Workloads in IaaS Clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 264–275, 2018.

[10] "AMPL: Streamlined modeling for real optimization," 2019, available at https://ampl.com/, last accessed on 10th Jul 2019.

[11] "OMNeT++ discrete event simulator," 2019, available at https://omnetpp.org/, last accessed on 2nd Sep 2019.

[12] R. Beraldi, H. Alnuweiri, and A. Mtibaa, "A Power-of-Two Choices Based Algorithm for fog Computing," *IEEE Transactions on Cloud Computing*, vol. 7161, no. c, pp. 1–12, 2018.

[13] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli, "Optimal operator placement for distributed stream processing applications," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, ser. DEBS '16. ACM, 2016, pp. 69–80.

[14] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[15] S. Svorobej, P. T. Endo, M. Bendechache, C. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn, "Simulating fog and edge computing scenarios: An overview and research challenges," *Future Internet*, vol. 11, no. 3, pp. 1–15, 2019.