Chapter

SMART CITIES IN THE FOG: CLEARING THE VISION OF INNOVATIVE SENSING APPLICATIONS

***Nicola Bicocchi*[*], *Claudia Canali*[†] *and Riccardo Lancellotti*[‡]**
Department of Engineering "Enzo Ferrari",
University of Modena and Reggio Emilia

Abstract

The new paradigm of smart cities is deeply intertwined with the development of large-scale sensing applications. An ever-growing amount of sensors are collecting data to support decision strategies for the management of the city services. Examples of such applications are traffic monitoring, autonomous driving, environmental sensing, real-time power/resource utilization metering. A traditional cloud-based approach for the deployment of such services is likely to suffer from performance and QoS problems due to the risk of congestion on the data center outbound links and due to high latency related to the geographic data exchange. An alternative paradigm to mitigate these problems is the fog computing, where a layer of intermediate fog nodes is placed between the sensors and the cloud data center to reduce the amount of data exchanges (through aggregation and filtering) and to host latency-critical

[*]E-mail address: nicola.bicocchi@unimore.it.
[†]E-mail address: claudia.canali@unimore.it.
[‡]E-mail address: riccardo.lancellotti@unimore.it.

services. The fog computing opens several new issues for the management and deployment of the services, especially if we consider that new applications may be dynamically deployed and also the infrastructure is subject to changes over time (e.g., adding and removing sensors and fog nodes). While this dynamic behavior can be supported by existing technologies such as containers, service orchestration frameworks, and micro-services, the fog paradigm exacerbates the problem of infrastructure and service coordination and management to the point where new solutions must be devised. The critical challenges that should be addressed by future fog infrastructures for smart cities lie in the area of service management, optimization of the infrastructure and automatic deployment of applications. In the present chapter, we discuss advantages and disadvantages of solutions for the management of smart city sensing applications, considering architectures, optimization models, algorithms for the service deployment, and the support for the applications life cycle.

Keywords: Smart cities, Fog Computing, Orchestration

1. INTRODUCTION

Global population is likely to peak around 2050 at approximately 9 billion people, about two-thirds of which will live in cities [7]. This growth in urban areas will eventually result in an increasing demand of infrastructures, resources, and services. The *smart city* concept has been conceived for addressing this increasing pressure towards the integration of infrastructure components and services in numerous industries such as urban management, autonomous driving, healthcare, and retail [5]. Overall, smart city applications share the common long-term goal of enhancing city-wide resources management by leveraging sensing capabilities, wireless and wired communications, and distributed computational resources. A report from the McKinsey Global Institute estimates their annual economic impact in 2025 between \$4 trillions and \$11 trillions[1].

Since their inception, smart city technologies have been mainly developed using small-scale, closed-world approaches. Nevertheless, at the current stage of development, large-scale infrastructures are emerging and exposing drawbacks and limitations of the technologies used for implementation. As an example, the *cloud computing* paradigm is largely adopted in enterprises because of

[1]https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world

its scalable and distributed data management scheme. However, current cloud data centers face challenges because of the combined effects of the amount of data reaching them, and the additional requirements of location awareness and low latency necessary for smart city application.

Figure 1. A conceptual representation of a Fog Computing architecture.

As an alternative, *fog computing* extends the cloud computing paradigm to run distributed applications throughout the network [2]. Specifically, in a fog computing infrastructure (as shown in Figure 1) a layer of fog nodes is placed on the network edge to host some specific tasks to be performed on the data produced by the distributed sensors. In contrast to the cloud, the fog not only performs latency-sensitive applications at the edge of network, but also performs latency-tolerant tasks efficiently at powerful computing nodes at the intermediate of network. At the top of the fog tier, large cloud data centers can be still used for heavy-weight analytics. A typical advantage of using an intermediate layer of fog nodes is to reduce the data volume through pre-processing, filtering, and aggregation tasks performed at the network edge to avoid the risk of high network utilization in the connections towards the cloud level that can determine poor performance.

Motivated by these considerations, in this chapter we describe the main traits of fog computing architectures and show how their adoption could be beneficial in different domains. More in detail, we start discussing the most mature smart city verticals currently being deployed and their general features in terms of computing and networking requirements. Then, we survey recent proposals of fog computing architectures highlighting key features and the kind of application they are best suited to run. Finally, we present a general purpose framework able to optimize the automatic deployment of smart cities applications over a fog computing infrastructure considering both the characteristics of the infrastructure and the applications. The rest of this chapter is organized as follows. Section 2 presents the main smart city vertical applications along with their key requirements. Section 3 describes the most popular frameworks for smart city applications while Section 4 discusses computational models used for programming distributed applications. Section 5 introduces a general purpose framework for the design and the deployment of smart city applications over a distributed fog computing infrastructure. Section 6 describes how to model the problem of deploying applications over a distributed infrastructure according to specific key requirements. Finally, Section 7 draws concluding remarks.

2. Smart City Verticals

In this Section we overview the most prominent smart city vertical applications being currently developed in different industries. In Section 2.6, the same applications are also discussed and categorized in light of the key requirements of the smart city scenario.

2.1. Urban

Dense populations combined with complex infrastructure create opportunities for improving billions of lives and generate businesses. Up to 80% of cities in the United States are expected to adopt connected technologies within the year 2025 and create about $1.5 trillion dollars in economic value[2]. The most significant applications can be grouped within the following categories.

Mobility management. Data collected from distributed sensors reveal patterns of public mobility and transportation. Mobility operators use these data to enhance the traveling experience, and eventually improve safety and punctuality. At the same time, solutions for personal vehicles can determine the

[2]https://www.forbes.com/sites/blakemorgan/2019/11/01/top-80-stats-about-a-future-customer-experience-shaped-by-technology/

number, location and the speed of moving vehicles. Furthermore, smart traffic lights can automatically alter the lights based on the current traffic situation. Using historical data, smart solutions for traffic management also predict traffic evolution over time to prevent potential congestion. Finally, smart parking solutions can predict whether the parking spots are occupied or available for creating dynamic parking maps.

Energy and waste management. With a network of smart meters, municipalities can provide citizens with cost-effective connectivity to utilities companies' IT systems. Smart connected meters can send data directly to a public utility over the network, providing it with reliable meter readings. Smart metering allows utilities companies to bill accurately for the amount of water, energy and gas consumed by each household and to monitor city-wide consumption patterns. Furthermore, waste management solutions help to optimize waste collection schedules by tracking waste levels, as well as providing route optimization and operational analytics.

Public safety. Cities and communities are managing an exceptionally diverse and growing inventory of video information sources, including municipal and commercial CCTV feeds, home surveillance systems, traffic management videos, in-car and body-worn video in law enforcement and individual video sources. As a consequence, the aggregate amount of video data that cities and communities must manage and analyze is rapidly increasing. Smart cities are turning to visual surveillance solutions to cope with the volume of information and data at their disposal and, most importantly, identify events or key pieces of information as distinct from data noise. More in detail, the deployment of smart technologies can provide surveillance with the capability of identifying objects, entities, events, attributes or patterns of behavior (including temporal and special events, either real time or post event) from large bodies of video and audio streams.

Environmental monitoring. Environmental sustainability includes investments in systems for constantly monitoring temperature variations, air and water quality, oceanic changes that impact marine life, greenhouse emissions, vegetation cover, and ice/glacier spread. Using smart sensors, cities are able to better understand and predict the impact of environmental changes on their various constituents. These solutions help city managers prepare and plan for a range of environmental and ecological phenomena.

2.2. Industrial

The interconnection of machines and their sensors to collect valuable data for real-time insights is already increasing productivity, quality, and worker safety. It is expected that more than 70% of IoT value will be captured by industrial markets—estimated to exceed $4.6 trillion by 2025[3]. As an example, leading institutions and firms in Europe, have developed and published the Reference Architecture Model Industry 4.0 (RAMI 4.0) [25] describing the key concepts and standards of Industry 4.0.

The industrial applications of the smart paradigm are currently not intended for substituting traditional automation applications. In other words, smart technologies are still not involved in control loops at the field level, where the industrial controllers deal with physical production and automation systems must have strictly bounded reaction times. On the contrary, smart applications are related to supervision, optimization, and prediction, i.e., the so-called digital manufacturing.

Digital manufacturing. The term digital manufacturing introduces digital technologies to promote the integration of product design processes, manufacturing processes, and general collaborative business processes across factories and enterprises [6, 17]. An important aspect of this integration is to ensure *interoperability* between machines, products, processes, and services, as well as any descriptions of those. Accordingly, a digital factory consists of a multi-layered integration of the information related to various activities along the factory and related resources.

2.3. Automotive

Connected vehicles capable of bidirectional communication with an external network for the purpose of delivering digital content and services, transmitting telemetry data from the vehicle are being developed at rapid pace.

[3]https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/

Autonomous vehicles. Autonomous vehicles hold the promise of reducing traffic fatalities, reducing congestion as well as curbing our carbon footprint. According to ABI, a marketing firm, roughly 8 million (10% of global output) vehicles with self-driving capabilities of level 3 or higher will be shipped in 2025[4].

Predictive maintenance. Connected cars and related city infrastructures also ensure implementation of fleet and vehicle health solutions by leveraging in-vehicle data collection, in-cloud data management, and user analytics. Furthermore, the concept of predictive maintenance is progressively transforming into a practical and straightforward solution integrating vehicle's sensors, hardware modules, data transmitters, and control units, allowing the tracking of performance, and eventual damage factors.

Infotainment Cloud-enabled platforms for value-added user services and applications are growing. Car manufacturers, maintenance and service companies, insurance companies, and entertainment providers can offer a broad range of innovative services for connected car.

2.4. Retail

In the retail industry, IoT and cloud technology are helping to improve prediction, precision and production in stores, warehouses and businesses. Remote monitoring, improved production and reduced waste are just a few benefits that IoT technology can provide to businesses and their projects. Being able to track and finalize a consumer's purchases, as well as their product history and specific demographics provides retail consumers with heightened convenience. This trend also benefits retailers through lower labor costs and an improved understanding of their audience. A list of significant retail-oriented applications can be summarized as follows.

Location-aware advertisement and navigation. Bluetooth beacons provide shoppers with both indoor and outdoor localization and information about shops nearby. Beacons also help vendors focus their marketing campaigns on actual customer behavior. On the same line, IoT-led data tracking allow retail firms to get advanced metrics on the flow of people within the stores and the

[4]https://www.goldmansachs.com/insights/technology-driving-innovation/cars-2025/

best selling points for individual customers.

Smart mirrors. Can be adopted within fitting rooms for suggesting other items based on what others have bought using an RFID label-scanning system. Smart mirrors have also been used in conjunction with augmented reality allowing customers to virtually dress themselves without having to do it physically.

Smart tags and packaging. Smart tags (tiny digital screens or labels) can replace traditional price tag allowing retailers to adjust pricing based on the store stock levels, peak periods and current sales. Finally, smart packaging that monitors the freshness of products and even smart refrigerators which monitor product inventory and restock themselves according to temperature would also be utilized within the store environment. This would make it a self-sufficient system, with a much lower demand for physical staff members, therefore saving on cost.

2.5. Healthcare

The global IoT in healthcare market size is projected to reach USD 534.3 billions by 2025 expanding at an annual growth rate of 19.9% over the forecast period 2019-2025[5]. Recent technologies have penetrated various fields of healthcare services, including chronic disease monitoring and management, home health, physical and occupational therapy, consumer and professional education, disaster management and even dentistry. In the following a list of applications related to the healthcare area is provided.

Medicare interaction. Personal health and medical data are collected from an individual in one location and afterwards are transmitted to a provider in a different location for use in care and related support. In this way the provider can track healthcare data for a patient once released to home or a care facility, reducing readmission rates. This approach can maintain individuals' health in their home and community, without making them physically go to the providers' office. Furthermore, the interactions between individuals and a provider can be aided with live video streams for both consultative and diagnostic and treatment services.

[5]https://www.reportlinker.com/p05763769/?utm_source=PRN

Health tracking. Health practice and education have been increasingly supported by mobile communication devices such as smartphones, tablets, or PDAs. Applications can range from targeted text messages for medicare, continuous monitoring of health parameters or conditions, to wide-scale alerts about disease outbreaks.

2.6. Requirements for Smart City Applications

The applications highlighted above, despite sharing the common ground of making use of IoT technologies and cloud services, are profoundly different in terms of use cases and, thus, computational and networking requirements. As an example, autonomous driving capabilities require strictly controlled latency margins, while medicare application offering video streaming are more likely to be constrained by bandwidth availability. The near coming transition between closed-world, independent applications and general purpose infrastructures capable of dealing with diverse use cases without intervention require hardware and software tools capable of finding adaptive tradeoffs among different requirements. In the following, we introduce a taxonomy based on 5 categories representing the majority of smart city vertical applications. Table 1 summarizes the results.

1. **Latency-bound**: applications with strong requirements in terms of response times, as in real time contexts. Examples of applications: mobility management, public surveillance, autonomous driving, and location aware advertisement and navigation belong to this category.

2. **CPU-bound**: applications characterized by computationally heavy tasks with high processing times where the CPU is likely to represent a bottleneck. Examples of applications: public surveillance, autonomous driving, infotainment systems, smart mirrors and medicare applications are in this category.

3. **Bandwidth-bound**: applications characterized by significant amount of data to be transferred, where the network bandwidth is key for delivering services. Examples of applications: public surveillance, digital manufacturing, autonomous driving, infotainment, smart mirrors and remote medicare interaction.

Table 1. Smart city applications represented in terms of their key constraints.

Application	Latency	CPU	Bandwidth	Reliability	Statefulness
Mobility management	●	○	○	○	●
Utility/Waste	○	○	○	○	○
Public surveillance	●	●	●	○	○
Digital manufacturing	○	○	●	●	○
Autonomous driving	●	●	●	●	○
Predictive maintenance	○	○	○	○	●
Infotainment	○	●	●	○	○
Location-aware Ads/Nav	●	○	○	○	○
Smart Mirrors	○	●	●	○	●
Smart packaging	○	○	○	●	○
Medicare interaction	○	●	●	●	○
Health tracking	○	●	○	●	○

4. **Reliability-bound**: applications where the reliability – intended as integrity, security, privacy – of data is a key requirement. Examples of applications: digital manufacturing, autonomous driving, smart packaging and medicare applications.

5. **Stateful**: applications needing to access past information for processing incoming data, hence data flows can not be distributed over multiple nodes. Typically, applications in which aggregated views of the data are needed. Examples of applications: mobility management, predictive maintenance, smart mirrors.

3. Smart City Architectures

Despite the relative lack of fog computing (FC) platforms, in this Section we discuss the most prominent frameworks supporting smart city applications in light of the categories of applications introduced in Section 2.6.

Recently, in [11] authors proposed approaches and architectures for enabling cloud-mediated interactions with nodes hosting either sensors or actuators. *Stack4Things* builds upon the *OpenStack* framework and aims at delivering Sensing-and-Actuation-as-a-Service (SAaaS) capabilities. It is focused on latency-bound applications, while enabling scalability of both CPU and bandwidth requirements.

In [21] authors propose to leverage high-level information-centric principles for IoT (i.e., resource pooling, content storing, and node locating) within the fog computing paradigm. Their results show that the achieved performance, regarding total packet number and average transmission latency, can outperform that of previous schemes.

In [13] authors propose an architecture for collecting, updating, and processing real-time and heterogeneous data from different sources and actors of the urban mobility scenario. Authors validated the proposed model by making use of map-based smart parking services and tested efficiency and accuracy and of the traffic monitoring algorithms using real-world traffic data. This platform is best suited for running latency-bound mobility applications.

The *Fog Data* service-oriented architecture [9] has been proposed for reducing cloud storage and subsequent delays in transmission for health-related applications. The architecture is organized in three tiers devoted to: *(i)* data collection via wearable sensors, *(ii)* data processing and filtering on fog nodes, and *(iii)* secondary, and most complex analysis on cloud data. Authors validate their results with two case studies involving a speech disorder and an ECG. Since its inception, this approach is targeted at CPU and bandwidth-bound applications typical of the healthcare industry.

More recently authors presented a networking prototype providing computing resources for the fog layer within the 5G network [19]. The architecture allows dynamic deployment and composition of virtual functions which are a key component of 5G networks. More in detail, they describe design and implementation of a streaming service making use of mobile edge computing features. The analysis of the case study gives a initial confirmation of the validity of the approach. However, despite its appeal, the architecture does not support highly dynamic deployment scenarios and does not account for latency requirements. Due to these characteristics, the approach is most likely suited for applications requiring bandwidth over a relatively stable network of nodes as for public surveillance and retail applications.

In another work [14], authors present innovative strategies for green networking and computing, such as server consolidation or energy aware routing, inside cloud data centers. The work is aimed at of protecting quality and service level against resource demand uncertainty while considering latency constraints on the fog nodes. It provides for robustness in case of unpredictable resource demand variations, powers down unused hardware, and calculates the optimal network embedding in light of latency constraints. Despite this proposal do not explicitly distinguish fog nodes and fails in addressing traffic management between fog nodes, it is well positioned to manage latency-bound applications with significantly variable loads such as mobility management and autonomous driving.

In [23], the authors focus on integrating intelligence in fog architectures for performing data representation, feature extraction, identifying outliers, and eventual hazardous events. They analyze different case studies for detecting events threatening pipeline safety. A working prototype has been constructed for evaluating the performance in recognizing 12 specific events. The results show the feasibility of city-wide, distributed machine learning platforms in the near future.

The above examples show how the IoT and Fog Computing concepts can help to realize the smart city vision and overcome the difficulties associated with cloud data centers. However, none of these works show how they can allocate and manage the appropriate resources at the Fog layer to further reduce delays and energy consumption. Starting from these consideration, we introduce in Section 5 a general framework for the automatic deployment of smart city applications over a fog infrastructure.

4. SMART CITY COMPUTATIONAL MODELS

Computational models for distributed systems, capable of abstracting from individual networked devices and working at the level of their collective behaviors, have prominent examples in SCEL [8], or SAPERE [4]. These also include space-time models for the universal manipulation of field data structures diffused in space and evolving in time, as in field-based [12] and aggregate computing [10]. The aim of these models is to enable the seamless modeling of small-scale composite services as well as large-scale collective services, and to support the transparent integration of semi-decentralized approaches with fully distributed ones.

In addition, frameworks to automatically split data processing behavior for cloud- and cluster-style execution (e.g., Spark [6] and Flink [7]), enrich traditional collective and aggregate approaches to distributed programming by considering a transition from handling collective *fields* of data to the notion of distributed stream of events, thus addressing also non-functional aspects concerning the control of dynamic aspects of event generation and diffusion.

Some recent approaches to programming IoT services propose new computational abstractions as building blocks of IoT services, such as the microservices of Osmotic computing [24], the core processors of EdgeIoT [22], or the deployment units of Elastic Computing [16]. These approaches share the idea of enabling the adaptive deployment of such building blocks at the level of both cloud or fog nodes, and possibly at the level of IoT devices. However, they are lacking a truly operational semantics dynamically addressing multi-tiered architectures, with the possibility of also acting in collective terms at the level of devices to enforce large-scale adaptive service composition.

Concerning middleware, a variety of platforms have been proposed to support the deployment and execution of IoT services and applications (see [18] for a survey) and including solutions to adaptively handle interoperability, context-dependency, spatial aggregation and adaptivity. These platforms promote interoperability, adaptivity, context-dependency and solutions for the collective execution and coordination of processes on the large-scale.

Concerning software engineering, it is recognized that the development of IoT systems and smart city applications may require not only the extension of existing methods but also novel methodologies and tools. However, at the time of writing methodologies and tools specifically suited to future IoT scenarios are still lacking, together with tools (novel or extensions of existing ones) in support of the verification of IoT system behaviour, and that can be of general use in the context of large-scale distributed systems.

5. A Flexible General Purpose Manager for Fog Computing Applications

To address the need for flexibility and adaptability in the design, configuration and deployment of a fog computing infrastructure for smart cities applications,

[6]https://flink.apache.org/

[7]https://spark.apache.org/

we propose the concept of a framework able to generate all the information necessary for automatic and optimized deployment of a specific application on a fog infrastructure. The core of the proposed framework is a *general purpose manager* able to take in consideration several elements depending on the fog infrastructure and on the main application characteristics, including the key requirements depicted in Section 2.6.

The presence of an intermediate layer of fog nodes opens new problems for the application deployment. For example, the mapping between the data flows coming from the sensors and the fog nodes that have to perform data (pre-)processing tasks is a critical aspect for the performance of the applications, as demonstrated by a recent study [3]. Moreover, different application requirements may have opposite consequences on the deployment: for example, for a latency-bound application the most important condition typically is to minimize the response time even at the price of some data loss, while for a reliability-bound application the data integrity is the fundamental requirement.

In the following of this section we describe in detail the main components of the proposed general purpose manager.

Figure 2. General Purpose Manager.

Fig. 2 presents the scheme of the general purpose manager, highlighting the presence of two main components: the *Orchestrator* and the *Deployer*. The Orchestrator is the responsible for the generation of the optimized configuration for the application deployment. This component receives three main inputs: the *infrastructure description*, the *application requirements* and the *current deploy-*

ment state. The first input of infrastructure description includes all the information characterizing each element involved in the distributed fog infrastructure: for each available fog node, the input information describes the physical position, the main characteristics and available resources (in terms of processing capabilities, bandwidth, etc.), and the software available on the node. Similar information are received for the cloud nodes in available cloud data centers that (eventually) receive the data processed by the fog nodes. The second input is about the requirements of the application to be deployed on the fog computing infrastructure. The requirements cover different aspects of the application, ranging from the number and physical location of the distributed sensors along with their main characteristics, to the information about the application data flow (e.g., sensor output data rate) reaching the fog infrastructure, up to the specific application constraints, such as the specific requirement(s) as categorized in Table 1 and the required SLA (Service Level Agreement). The final input to the Orchestrator is the state describing the current deployment of the application on the fog infrastructure; this state includes the mapping of the data flows coming from the sensors on the fog nodes responsible for the data processing.

The Orchestrator exploits different *Plugins* to take in consideration the requirements of the application to be deployed. Each plugin addresses one of the specific requirements described in Section 2.6 through the generation of an optimized deployment of the application on the fog infrastructure. The output of the Orchestrator is a set of *commands for deployment*, that is received by the *Deployer* component.

The *Deployer* processes the information about the application deployment and produces three output files that contain the specific and local commands for the automatic deployment on each element involved in the fog infrastructure: the sensor nodes producing data, the fog nodes and the cloud nodes. Specifically, the Deployer produces as output .app bundles/packages actually containing a hierarchy of files and folders with executable programs, other software components and data.

In the following section, the details about how the Plugins can address the specific requirements of the applications are described.

6. QUANTITATIVE MODELING

In this section, we model the general problem of deploying a smart city application over a distributed fog infrastructure as the one shown in Fig. 1. Moreover, we detail the constraints and the specific conditions to be considered in the model to address the application key requirements described in Section 2.6.

In our model, we assume a scenario where a set of geographically distributed sensors $\mathcal{S}$ produce data at a steady rate: we denote with λ_i the frequency for the generic sensor i. It is worth to note that we consider static sensors, but a scenario where mobility is taken into account can be easily introduced in our model. The fog layer consists of a set of nodes $\mathcal{F}$ that receive the data from the sensors and performs operations on them. As discussed in Section 2, these operations may include, for example, real-time data processing for automatic driving support, data aggregation for traffic monitoring, and identification of anomalies for predictive maintenance or public safety. The refined data samples from the fog nodes are then sent to the cloud where additional analysis is carried out and all the information is stored: specifically, in our general model we consider a set of cloud data centers $\mathcal{C}$.

To address the requirements of a smart city application, we need to guarantee the expected QoS. To this aim, first of all we have to consider the application response time and its three main contributions:

1. **Network-based latency between sensors and fog layers** This latency is due to the communication from the sensor to the fog nodes: we denote this value as $\delta_{i,j}$ where i is a sensor and j is a fog node.

2. **Network-based latency between fog and cloud layers** This latency is due to the communication from the fog nodes to the cloud data centers: we denote this value as $\delta_{j,k}$ where j is a fog node and k is a cloud data center.

3. **Computation time on the fog node** This time depends on the computation cost of the request (we denote as $1/\mu_j$ the time to process a packet of data from a sensor on fog node j) and on the data rate λ_i of all the sensors i that are communicating with the fog node j.

As the problem concerning the management of large cloud data centers has been widely addressed in literature [20], we do not consider the inner details of the cloud layer in our problem modeling, such as the computation time at the level of the cloud data center, but we just focus on the steps occurring before reaching the cloud layer.

To determine the efficient deployment of a smart city application over the fog infrastructure, the orchestrator component of the general purpose manager

Table 2. Notation

Symbol	Meaning/Role
$\mathcal{S}$	Set of sensors
$\mathcal{F}$	Set of fog nodes
$\mathcal{C}$	Set of cloud data centers
λ_i	Outcoming 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 to fog node j
$\delta_{j,k}$	Communication latency between fog node j to cloud data center k
b_j	Bandwidth available at the fog node j
$x_{i,j}$	Sending data flow from sensor i to fog node j
$y_{j,k}$	Sending data flow from fog node j to cloud data center k
i	Index of a Sensor
j	Index of a Fog Node
k	Index of a Cloud Data Center

has to solve the problem of mapping the data flows coming from the sensors over the fog nodes [3]. To this aim, we define an optimization problem following an approach similar to the problem of allocating resources over a distributed infrastructure, such as VMs on a Cloud data center [20, 15]. Specifically, we use a matrix X of decision variables $x_{i,j}$ determining if sensor i is sending data to fog node j. Similarly, we use a matrix Y of variables $y_{j,k}$ to indicate that the fog node j sends data to the cloud data center k. The reader may refer to Table 2 for a complete summary of the parameters used in our model.

The general optimization problem is defined below.

$$\min obj(X,Y) \tag{1}$$

subject to:

$$\sum_{j\in\mathcal{F}} x_{i,j} = 1 \quad \forall i \in \mathcal{S}, \tag{2}$$

$$\lambda_j < \mu_j \quad \forall j \in \mathcal{F}, \tag{3}$$

$$0 \le x_{i,j} \le 1, \tag{4}$$

$$0 \le y_{j,k} \le 1, \tag{5}$$

The specific objective function to be minimized depends of the peculiar characteristics of the smart cities application to be deployed. Depending on the specific key requirement(s) of the application, the orchestrator relies on the corresponding plugin(s) to define the correct optimization problem: the main idea is that each application key requirement needs a different objective function 1 and/or specific constraints to be addressed, as detailed in the rest of this section. Moreover, the general optimization problem has two constraints. Constraint 2 guarantees that the data flow coming from a sensor i is partitioned on the fog nodes. Constraint 3 guarantees that, for every fog node j, we avoid a congestion situation, where the incoming load at a fog node j exceeds the processing capability μ_j of that node. To this aim, we denote λ_j as the incoming data rate at the fog node j, defined as follows:

$$\lambda_j = \sum_{i \in \mathcal{S}} x_{i,j} \cdot \lambda_i \quad \forall j \in \mathcal{F} \tag{6}$$

Latency-Bound Plugin

Latency-bound applications have strong requirements in terms of response times. Hence, in this case the objective function 7 to be considered in the problem formalization aims at minimizing the total (and hence the average) latency of the data flows going from the sensors to the fog nodes and from them to the cloud data centers. The objective function captures effectively the communication delay of a geographically distributed infrastructure using the latencies $\delta_{i,j}$ and $\delta_{j,k}$.

$$obj(X,Y) = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} x_{i,j}\delta_{i,j} + \sum_{j \in \mathcal{F}} \sum_{k \in \mathcal{C}} y_{j,k}\delta_{j,k} \tag{7}$$

CPU-Bound Plugin

CPU-bound applications are characterized by computationally heavy tasks with high processing times where the CPU is likely to represent a bottleneck. Hence, in this case the objective functions aims to minimize the processing time of the operations carried out at the level of the fog nodes, as expressed in Equation 8. The expression of the processing time used for our objective function is consistent with other studies in literature focusing on distributed cloud infrastructures [1]. Specifically, the average processing time is derived from Little's result applied to a M/G/1 model and considers just the average arrival frequency λ_j and the processing rate μ_j of each fog node j.

$$obj(X, Y) = \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} x_{i,j} \cdot \Big(\frac{1}{\mu_j - \lambda_j} \Big) \tag{8}$$

In this case, we do not add the network delay contribution because in case of CPU-bound application it can be considered negligible with respect to the processing time and the main objective of the mapping is to balance the load among the fog nodes of the infrastructure.

Bandwidth-Bound Plugin

Bandwidth-bound applications are characterized by significant amount of data to be transferred over the network. In this case, we should guarantee that the data flow coming from the sensors assigned to a given fog node do not exceed its maximum bandwidth. To this aim, we denote with b_j the maximum bandwidth available at the fog node j and add to the optimization problem the following constraint.

$$\lambda_j < b_j \quad \forall j \in \mathcal{F} \tag{9}$$

It is worth to note that we do not consider as an issue the bandwidth between fog nodes and cloud data centers because we assume that fog nodes are placed in well connected locations. However, to consider a different case it is sufficient to add to the optimization problem a constraint similar to 9 regarding the data flow between fog nodes and cloud data centers.

Reliability-Bound Plugin

In case of reliability-bound application we have to guarantee the reliability - intended as integrity, security, privacy - of the transferred data as a key requirement. To this aim, we assume to have a probability of failure associated to each fog node and a SLA that defines the maximum acceptable probability of failure for the application to be deployed.

If we consider the set S of sensors associated to a reliability-bound application, and define $P_F(j)$ as the failure probability of a fog node j, the failure probability associated to the application con be expressed as

$$P_F(S) = 1 - \prod_{i \in \mathcal{S}} \left(1 - \sum_{j \in \mathcal{F}} x_{i,j} P_F(j) \right), \tag{10}$$

where j is a fog node receiving data from a sensors i belonging to S.

We can simplify the definition as

$$P_F(S) \approx \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{F}} x_{i,j} P_F(j), \tag{11}$$

if $P_F(j) \ll 1 \, \forall j \in \mathcal{F}$.

Given that, the constraint to be added to the optimization problem for a reliability-bound application is the following:

$$P_F(S) \leq P_{SLA} \tag{12}$$

Stateful-Bound Plugin

In the case of a stateful-bound application, the processing of current data need to access information on past results (as when aggregated views of data are needed), hence the data flows can not be distributed over multiple fog nodes. In our model it is sufficient to consider the decision variables $x_{i,j}$ as boolean values, meaning that $x_{i,j} = 1$ if and only if sensor i is sending all data to fog node j, otherwise $x_{i,j} = 0$. Basically, we have to add to the optimization problem the constraint 13 that defines the boolean nature of the decision variables $x_{i,j}$: this constraint guarantees that all the data of a sensor must be sent to the same fog node and cannot be distributed across the fog layer.

$$x_{i,j} = \{0, 1\}, \quad \forall i \in \mathcal{S}, j \in \mathcal{F}, \tag{13}$$

Multiple Requirements

Depending on the specific requirement(s) of the smart city application to be deployed, the optimization problem is defined by activating the corresponding plugin(s). In case of an application characterized by multiple requirements, the plugins contributions should be combined. For example, let us analyze the case of an application of public surveillance. As can be observed in Table 1, the application is both latency- and CPU-bound: in this case, the two objective functions defined by the corresponding plugins (7 and 8) can be joined through a linear combination with weights determined by the specific application context. Moreover, the application is also bandwidth-bound: hence, the constraint 9 has to be added to the optimization problem.

7. Conclusion

Modern smart cities applications increasingly rely on geographically distributed sensors that produce data to be processed with different goals, giving origin to scenarios characterized by severe (and sometimes opposite) requirements in terms of latency, CPU, bandwidth, reliability, and statefulness. While Fog Computing may represent an immediate solution to reduce network traffic and improve latency by placing computational resources at the edge of the network, this new paradigm opens new issues in the deployment of a smart cities application. Indeed, the presence of a distributed intermediate layer arises several possibilities for the mapping of the data flows coming from the sensors over the fog nodes, that becomes a critical aspects to guarantee all the specific requirements of modern applications.

In this chapter, we analyze the most popular smart cities vertical applications to understand their key features and requirements. Moreover, we consider some recent proposals of fog computing infrastructures and discuss how these requirements impact on the supporting distributed architectures.

We also propose a general framework allowing the automatic deployment of a distributed smart cities application, based on the specific application requirements. The core element of framework is a component functioning as an orchestrator, mapping the sensors over the distributed fog nodes as an optimization problem and solves it. Through the use of dedicated plugins, the optimization problem is automatically adapted to address the specific key requirements of the application to be deployed over the fog infrastructure.

Starting from some real smart cities scenarios, the proposed solution shows how the specific application requirements can be considered and eventually combined to reach a correct deployment over a fog computing infrastructure in order to satisfy the expected SLA. The proposed general purpose manager represents a flexible instrument to support the planning and the dynamic management of modern smart cities applications.

REFERENCES

[1] Ardagna D., Ciavotta M., Lancellotti R., and Guerriero M.. A hierarchical receding horizon algorithm for qos-driven control of multi-iaas applications. *IEEE Transactions on Cloud Computing*, pages 1–1, 2018.

[2] Bonomi Flavio, Milito Rodolfo, Zhu Jiang, and Addepalli Sateesh. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[3] Canali Claudia and Lancellotti Riccardo. 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.

[4] Castelli Gabriella, Mamei Marco, Rosi Alberto, and Zambonelli Franco. Engineering pervasive service ecosystems: the sapere approach. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(1):1, 2015.

[5] Chourabi Hafedh, Nam Taewoo, Walker Shawn, Gil-Garcia J. Ramon, Mellouli Sehl, Nahon Karine, Pardo Theresa A., and Scholl Hans Jochen. Understanding smart cities: An integrative framework. In *2012 45th Hawaii international conference on system sciences*, pages 2289–2297. IEEE, 2012.

[6] Chungoora Nitishal, Young Robert I., Gunendran George, Palmer Claire, Usman Zahid, Anjum Najam A., Cutting-Decelle Anne-FrançOise, Harding Jennifer A., and Case Keith. A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry*, 64(4):392–401, 2013.

[7] Davis Kingsley. The urbanization of the human population. In *The city reader*, pages 43–53. Routledge, 2015.

[8] De Nicola Rocco, Latella Diego, Lafuente Alberto Lluch, Loreti Michele, Margheri Andrea, Massink Mieke, Morichetta Andrea, Pugliese Rosario,

Tiezzi Francesco, and Vandin Andrea. The scel language: Design, implementation, verification. In *Software Engineering for Collective Autonomic Systems*, pages 3–71. Springer, 2015.

[9] Dubey Harishchandra, Yang Jing, Constant Nick, Amiri Amir Mohammad, Yang Qing, and Makodiya Kunal. Fog data: Enhancing telehealth big data through fog computing. In *Proceedings of the ASE bigdata & socialinformatics 2015*, page 14. ACM, 2015.

[10] Jelasity Márk, Kowalczyk Wojtek, and Van Steen Maarten. An approach to massively distributed aggregate computing on peer-to-peer networks. In *12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2004. Proceedings.*, pages 200–207. IEEE, 2004.

[11] Longo Francesco, Bruneo Dario, Distefano Salvatore, Merlino Giovanni, and Puliafito Antonio. Stack4things: a sensing-and-actuation-as-a-service framework for iot and cloud integration. *Annals of Telecommunications*, 72(1-2):53–70, 2017.

[12] Mamei Marco and Zambonelli Franco. *Field-based coordination for pervasive multiagent systems*. Springer Science & Business Media, 2006.

[13] Marchetta Pietro, Natale Eduard, Pescapé Antonio, Salvi Alessandro, and Santini Stefania. A map-based platform for smart mobility services. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 19–24. IEEE, 2015.

[14] Marotta Antonio, D'Andreagiovanni Fabio, Kassler Andreas, and Zola Enrica. On the energy cost of robustness for green virtual network function placement in 5g virtualized infrastructures. *Computer Networks*, 125:64–75, 2017.

[15] Noshy M., Ibrahim A., and Ali H.A., Optimization of live virtual machine migration in cloud computing: A survey and future directions. *Journal of Network and Computer Applications*, 110:1–10, 2018.

[16] Nurmi Daniel, Wolski Rich, Grzegorczyk Chris, Obertelli Graziano, Soman Sunil, Youseff Lamia, and Zagorodnov Dmitrii. Eucalyptus: A technical report on an elastic utility computing architecture linking your programs to useful systems. In *UCSB Technical Report*. Citeseer, 2008.

[17] Papazoglou Michael P., Smart connected digital factories - unleashing the power of industry 4.0 and the industrial internet. In *Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018.*, pages 260–271. SciTePress, 2018.

[18] Razzaque Mohammad Abdur, Milojevic-Jevric Marija, Palade Andrei, and Clarke Siobhán. Middleware for internet of things: a survey. *IEEE Internet of things journal*, 3(1):70–95, 2015.

[19] Salsano Stefano, Chiaraviglio Luca, Blefari-Melazzi Nicola, Parada Carlos, Fontes Francisco, Mekuria Rufael, and Griffioen Dirk. Toward superfluid deployment of virtual functions: Exploiting mobile edge computing for video streaming. In *2017 29th International Teletraffic Congress (ITC 29)*, volume 2, pages 48–53. IEEE, 2017.

[20] Shojafar Mohammad, Canali Claudia, and Lancellotti Riccardo. A Computation- and Network-Aware Energy Optimization Model for Virtual Machines Allocation. In *Proc. of International Conference on Cloud Computing and Services Science (CLOSER 2017)*, Porto, Portugal, Apr. 2017.

[21] Song Fei, Ai Zheng-Yang, Li Jun-Jie, Pau Giovanni, Collotta Mario, You Ilsun, and Hong-Ke Zhang. Smart collaborative caching for information-centric iot in fog computing. *Sensors*, 17(11):2512, 2017.

[22] Sun Xiang and Ansari Nirwan. Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54(12):22–29, 2016.

[23] Tang Bo, Chen Zhen, Hefferman Gerald, Pei Shuyi, Wei Tao, He Haibo, and Yang Qing. Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Transactions on Industrial informatics*, 13(5):2140–2150, 2017.

[24] Villari Massimo, Fazio Maria, Dustdar Schahram, Rana Omer, and Ranjan Rajiv. Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Computing*, 3(6):76–83, 2016.

[25] Zezulka F., Marcon P., Vesely I., and Sajdl O., Industry 4.0 – an introduction in the phenomenon. *IFAC-PapersOnLine*, 49(25):8 – 12, 2016. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.