Taming the IoT Data Deluge: An Innovative Information-Centric Service Model for Fog Computing Applications

Mauro Tortonesi[1,*], Marco Govoni[1], Alessandro Morelli[1,3], Giulio Riberto[1], Cesare Stefanelli[1], Niranjan Suri[2,3]

1) Department of Engineering
University of Ferrara
Ferrara, Italy
name.lastname@unife.it

2) US Army Research Laboratory
Adelphi, MD, USA
niranjan.suri.civ@mail.mil

3) Florida Institute for Human & Machine Cognition
Pensacola, FL, USA
{amorelli,nsuri}@ihmc.us

*) Corresponding author:

Name: Mauro Tortonesi
Address: Department of Engineering
University of Ferrara
Via Saragat, 1
44122 Ferrara
Italy
Phone: +39 0532 974888
Fax: +39 0532 974888
E-Mail: mauro.tortonesi@unife.it

Taming the IoT Data Deluge: An Innovative Information-Centric Service Model for Fog Computing Applications

Abstract

Fog Computing is a new computation paradigm, recently emerged from the convergence of IoT, WSN, mobile computing, edge computing, and Cloud Computing, which is particularly well suited for Smart City environments. Fog Computing aims at supporting the development of time-sensitive, location-, social-, and context-aware applications by using computational resources in close proximity of information producers and consumers, such as increasingly common cheap and powerful modern hardware platforms. However, realizing Fog Computing solutions for Smart Cities represents a very challenging task, because of the massive amount of data to process, the strict resource and time constraints, and the significant dynamicity and heterogeneity of computation and network resources. These formidable challenges suggest taking into consideration new information and service model solutions that explore several trade-offs between processing speed and accuracy. Along these guidelines, we designed the SPF Fog-as-a-Service platform, which proposes a new information-centric and utility-based service model and allows the definition of self-adaptive and composition-friendly services, which can execute either on edge devices or in the Cloud. In numerous evaluations, SPF proved to be a very effective platform for running Fog services on heterogeneous devices with significantly different computational capabilities while also demonstrating remarkable ease of development and management characteristics.

Keywords: Fog Computing; Smart Cities; Information-Centric Networking; Internet-of-Things; Value of Information.

1. Introduction

Smart Cities represent massively heterogeneous, dynamic, and diverse environments, characterized by the ubiquitous presence of IoT devices producing a deluge of sensing data and by a myriad of IT applications that process this data in order to offer real-time and time-critical, location-, social-, and context-aware applications. Examples include emergency and health-care-related services [1], surveillance [2], augmented reality, gaming, social-oriented services [3] [4], effective traffic management [5] [6], etc. We expect these environments to become more and more prevalent following the deployment of an increasing number of sensors, new Smart City platforms, and the first 5G communication infrastructures.

In Smart Cities, traditional analytics solutions based on the transfer of all the data to the Cloud, processing using Big Data methodologies and tools, and returning the results to interested users [7] [8] are too slow for applications with strict latency constraints and too burdensome for the network infrastructure [9] [10]. Instead, the Smart City scenario is particularly well suited for the adoption of distributed processing approaches, such as Fog Computing.

Fog Computing is a relatively recent paradigm (one of the first manuscripts mentioning it is Bonomi et. al's in 2012 [11]), emerged from the convergence of several innovative trends in computing and communications, such as Internet-of-Things (IoT), M2M communications, Wireless Sensor Networks (WSNs), mobile computing, edge computing, and Cloud Computing [12] [13] [14]. As of today, and despite several attempts having been made [15] [10] [16], there is still neither a clear cut and universally accepted definition of Fog Computing, nor any (either de iure or de facto) standard service and/or programming models. However, researchers seem to

agree on the fundamental functions of the Fog as a computing platform to run information-processing services *at the edge of the network*, that is, on top of *edge devices* in proximity of either raw data sources, information consumers, or both.

However, the implementation of Fog Computing-based solutions for Smart Cities represents a very difficult task for three main reasons. First and foremost, processing a massive amount of data with scarce computational, bandwidth, and energy (IoT devices are often battery operated) resources available represents a formidable challenge. Cisco Systems Inc. recently predicted that by 2020 IoT devices will generate data at a rate as high as 600 ZB per year [17]. In addition, in this scenario users typically perform a dual role: not only do they access the information and services offered by the Smart City IT infrastructure, but the ever increasing number of smart personal devices (smartphones, tablets, and wearables) that they carry and the remarkable developments in their sensing capabilities means that citizens also effectively operate as information producers. In fact, on top of at least two cameras and a microphone, modern smartphones feature an iris scanner, a pressure sensor, a fingerprint reader, an accelerometer, a gyroscope, a barometer, a proximity sensor, a compass, a heart rate sensor, a light sensor, and a magnetometer. We can expect this to increase the burden on the network and on the computing infrastructures even more, and by a significant amount.

Second, edge resources are very dynamic and heterogeneous. In Smart City environments, IoT devices are often deployed in groups, e.g., sensing systems, and connected to the Internet through one or more "IoT gateway" devices, that typically have significantly better computation, storage, and communication capabilities. In addition, the functions provided by IoT gateways are being increasingly complemented by the deployment of "support" hardware in their proximity based on heterogeneous computing platforms [18]. The remarkable capabilities and (relatively) low power consumption of these platforms enable the execution of sophisticated and computationally hungry services while remaining fairly energy efficient. Finally, the ad hoc deployment of micro-clouds as impromptu edge computing data centers paves the way to dynamic resource exploitation and interesting smart service development opportunities [19].

Third, communications are also very heterogeneous [20]. In addition to the multitude of short-range and low-power wireless communications protocols, e.g., IEEE 802.15.4, Bluetooth LE, NFC, etc., used in IoT networks, a plethora of different medium-range wireless protocols, e.g., WiFi, WiFi-direct, LTE/4G, LTE-direct, etc., are commonly adopted for connecting personal devices to the network infrastructure, IoT gateways, or other personal devices. On top of that, significant mobility of users and terminals causes frequent communication disruptions and wide variability in channel performance.

To address these challenges and enable matching the low latency requirements of a new generation of deeply immersive and high-value-added services designed for the citizens of the Smart Cities of the future, Fog Computing based solutions require smart information and resource management methodologies and tools. More specifically, there is a need for solutions that can exploit the heterogeneous and ever changing pool of edge devices for service execution in a dynamic and efficient fashion and prioritize the allocation of resources for (raw) data analysis and (processed) data dissemination to highly important information, even at the cost of ignoring or discarding lesser important information. At the same time, there is a need to coordinate information and resource management tools between Fog and Cloud in order to

transfer to the Cloud (the portions of) information that would be too computationally expensive to process in the Fog.

2. Fog Computing in Smart Cities

The peculiar characteristics and requirements of the Smart City scenario raise significant opportunities and challenges for the design of Fog Computing platforms [21]. In fact, the realization of real-time and time-critical, location-, social-, and context-aware applications capable of fully reaping the enormous potential of Fog Computing requires innovative methodologies and tools for service development, management, and deployment that go well beyond the limits of current state-of-the-art solutions, often designed for Big Data analytics in Cloud computing environments.

First of all, the formidable deluge of raw data to analyze and the strict resource and time constraints suggest the opportunity to consider new solutions at the information and service model level that explore several trade-offs between processing speed and accuracy [22]. To this end, a first step lies in the adoption of an information model that explicitly considers the differences between the many types of data that are processed and exchanged in this scenario, e.g., raw data generated by sensors, aggregated raw data, processed information ready for consumption from a human or software component, and so on. Additionally, such model could be enriched by taking into account the specific characteristics of each information object in terms of generation time, redundancy, relevance, set of recipients, and so forth.

In this context, *Value-of-Information (VoI)* represents a particularly attractive metric. The birth of the VoI concept, which measures the utility of information according to a subjective and consumer-centric perspective, can ultimately be traced back to the seminal work by Howard [23]. His study attempted to extend Shannon's information theory to consider "the probabilistic nature of the uncertainties that surround us, but also with the economic impact that these uncertainties will have on us". The investigation of the utility that each discrete element of information provides to its consumer(s), which has been an active research topic in economic and decision making theories for the last 50 years and is still receiving a considerable amount of attention [24] [25], holds interesting promises for several application scenarios, including Fog and Cloud computing [26] [27].

In fact, classifying information according to the value it provides to its recipients represents a very natural and effective criterion for "pruning" the share of data whose processing is not allowed by the limited amount of Fog resources available. For instance, sensing data that do not add significant value to the knowledge already built from the analysis of previous sensor readings would be characterized by a low and quickly decreasing VoI, and thus have very low probability of being selected for processing in (the very likely) case of resource shortages.

The systematic adoption of solutions to discard data with lesser importance also allows to consider self-adaptive models for Fog Computing services. Services could be designed to be constantly aware of the resources available in their current location of execution and in the surrounding environment and capable of prioritizing the processing and delivery of essential information, possibly at the expense of non-essential information, in case of shortages. This

approach provides significant advantages in terms of QoS tailoring in light of the highly varying (bandwidth and computational) resource availability in Smart City scenarios.

The information-centric perspective would also allow the system to better cope with the non-trivial communication challenges that Fog Computing applications need to address in these scenarios [20]. In fact, the communication requirements of Smart City environments call for resilient information dissemination solutions that can withstand network disruptions and performance issues while, at the same time, leveraging multiple interfaces and user mobility to support communications [28]. The adoption of Information-Centric Networking (ICN) inspired solutions in place of more traditional end-to-end communications would naturally enable taking advantage of multiple network interfaces in a heterogeneous environment, to withstand channel disruptions and network partitions, and to leverage mobile users as "message ferries" to improve communications. Their high effectiveness in disseminating transient information and dynamically generated content makes ICN solutions with publish-subscribe semantics particularly appealing for supporting Smart City applications and the time sensitive, location-aware, and social-aware services they build on top of.

At the same time, the adoption of utility-based prioritization solutions would also bring significant benefits to the dissemination of processed information, a topic that has unfortunately received less attention from scientific literature so far. In fact, the prioritization of highly valuable information in data dissemination represents a natural and very effective criterion for optimal bandwidth exploitation [26].

Finally, the information-centric perspective also provides substantial advantages in service composition, which is another very challenging and often overlooked problem in Fog Computing. In fact, many IoT applications and services allow the simultaneous processing of the same information in different ways, according to the current application and user contexts [29]. In Smart City environments, the composition of Fog Computing services is further challenged by the limited bandwidth resources and connectivity issues and by the mobility of services that could be dynamically migrated (for example from the Fog to the Cloud in case more processing power is required). The definition of composed services based on the kind of information they consume and produce, as opposed to solutions such as WSDL and BPEL that consider the service API and location instead, represents an approach that is more resilient to communication problems and service migrations.

At the resource management level, there is the need for Fog Computing solutions to consider many different devices and platforms as possible locations for service execution. IoT gateways, heterogeneous computing platforms, and micro-Clouds are all very interesting candidates for the execution of information processing tasks. Unfortunately, all these options come with significant trade-offs: IoT gateways present non-trivial constraints in terms of computation, energy, and communication resources, heterogeneous platforms present an increased programming complexity that requires the extensive and time-consuming rewrite of applications using specialized programming paradigms, and micro-Clouds need to be supported by robust and dynamic resource discovery and management solutions.

In addition, there is a need to consider solutions that can manage the ever-changing pool of resources in Fog Computing environments, discovering information sources, monitoring resource availability, and reacting to changes according to the service requirements and the users' preferences. Also, we can envision that most of these resources will be available in the

context of a public platform operated by a Fog provider – a new kind of player that will likely emerge as a key role in the Smart City environments of the near future – offering pay-per-use computing functions on top of a wide variety of different edge devices or in the Cloud through a proprietary API. This means that Fog Computing solutions for Smart City environments will be a heavily heterogeneous and fragmented scenario not just from the hardware perspective, but also from the administrative one.

These considerations call for comprehensive Fog Computing solutions capable of implementing coherent and homogeneous management functions for a plethora of different services running on diverse but federated Cloud and Fog environments. These solutions should support the allocation of services on either a suited edge device or the Cloud, according to the service requirements and the current network conditions. An even more ambitious goal would be to dynamically and automatically implement the migration of services between the Cloud and the Fog, possibly in reaction to the detection of an unexpected activation of new resources or to the impromptu increase in request workload, thus realizing a service continuum that has being recently proposed as a pillar of future Fog and IoT research [14].

The resource scarcity of the Fog suggests considering different models for running services. In addition to "resident" (i.e., long-time running, services), there is a need to develop services that can be activated on demand and for a short amount of time, automatically being deactivated when not needed [4]. We call these services "normally-off", as they respond to the same issues raised by researchers working on normally-off computing [30], and represents a similar trend at the software engineering level.

3. The SPF Information-Centric and VoI-based Information Model and Management

We realized a Fog-as-a-Service platform, called SPF (as in “Sieve, Process, and Forward”), to address the issues of Smart City environments. SPF proposes an innovative information model and a corresponding information-centric and value-based service model to deal with the main challenge of Smart City environments, i.e., processing the deluge of raw data generated.

To illustrate the SPF information and service models, let us consider how our platform could enable the development, deployment, and management of Fog services in support of citizens’ during a holiday celebration in a Smart City environment. In this scenario, a section of the city center will be closed to automobile traffic and will be accessible only to pedestrians. In response to the gathering of a large crowd, Police forces and EMS personnel will be deployed to guarantee order and safety for the citizens. Information sources, such as traffic cameras, usually dedicated to day-to-day monitoring of the city, will capture data feeds that will be funneled into Fog Computing platforms for analysis.

Several applications will run concurrently in the Smart City Fog Computing platform, each one leveraging a set of information processing services. First of all, a logistics / organization support application will leverage a crowd monitoring service to analyze the number of personal devices connected to the network and video camera feeds to provide a relatively accurate and up-to-date estimate of the number of people present in the city center. This information will help to right

size the team of EMS personnel and resources (public water services, hygiene spots, ambulances, etc.).

In addition, a security check application, running with the highest possible priority and with no associated resource consumption policies / limits, will continuously scan video feeds collected from, e.g., traffic cameras, for anomalies. Such an application might be developed to take advantage of both the low latency processing allowed by the Fog and the computational capabilities of the Cloud. More specifically, the application will implement a first-order security control service in the Fog that runs coarse-grained anomaly detection algorithms with relatively light computational requirements on the data collected by IoT devices. Its goal is to help police forces to identify as quickly as possible obvious and/or evident security threats such as a person wielding a weapon or a group of people starting a brawl. At the same time, the application might implement a second order security control service in the Cloud, running fine grained face recognition algorithms against a database of known people, in order to help police forces to identify less evident and potentially more dangerous security risks, e.g., for counter-terrorism purposes.

Other applications might leverage Fog Computing environments to support EMS personnel in delivering medical services. For instance, an e-health application might try to identify possible health emergencies, such as heat strokes and dehydration, through the fusion of data collected from IoT (video feeds, temperature sensors, etc.), from wearable devices (heartbeat, sweat, and other physiological sensors), and from mobile devices (pace monitors, fall detector apps, and so forth).

Finally, public service applications and commercial applications can be run on the Fog Computing platform. Public service applications can provide useful information to citizens by facilitating the dissemination of traffic information or suggesting which metro train to get quickly outside of the city center. Other applications can provide services that integrate with IoT sensors, identifying impromptu performances from street artists or particularly interesting shopping sales, and directing users to their locations.

These applications operate on a wide variety of data types and present different requirements for the information processing tasks. In the rest of this Section, we will describe how the SPF information and service model can facilitate the development of these applications on top of a very heterogeneous and resource constrained scenario.

3.1. Information and Service Model

SPF proposes an information model that divides data in three different categories, according to the corresponding processing stage. First, there are *raw data*: input feeds of (typically sensing) data collected from WSNs or personal devices such as smartphones and wearables. Then, there are *Information Objects (IOs)*: higher-order constructs produced from a first level analysis of raw data, typically implementing some basic and/or generic processing function. IOs contain sensing information aggregated, “fused”, or distilled in smaller portions, which is typically much more valuable than raw data but not quite ready for consumption by end users. Finally, there are *Consumption Ready Information Objects (CRIOs)*: objects that contain fully-processed information in a format suitable to be returned to users as a response to their requests.

The SPF service model is also heavily information-centric and based on many different and composable processing layers. As illustrated in Fig. 1, *Fog services* take as input raw data generated by sensors and implement the analytics required to return a service response to users in terms of a CRIO. The processing function of a Fog service is implemented through the coordinated efforts of (at least) two different processing layers: *pipelines* and *services*. Pipelines analyze raw data to produce IOs. Services further analyze the pipeline-generated IOs by implementing application-specific processing, thus generating CRIOs.

This architecture maximizes the opportunities for reuse of processing components and of generated results. In fact, pipelines can (and are typically designed to) be used by many different services. They will typically implement basic functions, e.g., OCR processing, designed to be preinstalled by Fog providers in most or even all possible locations for service instantiation, i.e., either in edge devices or in the Cloud. Instead, services will typically implement application-specific tasks and will thus need to be purposely implemented by Fog application developers.

The relation between pipelines and services in our model can change dynamically to adapt to changes in the number and type of requests received, the services requested, the addition/removal of new resources or processing elements (Fog services, services, and pipelines), and so on. This gives SPF enough degrees of freedom to implement flexible dynamic service composition functions.

Note that not every raw data message leads to the generation of an IO, nor every IO generates a CRIO. In fact, IOs can emerge from the fusion of many raw data objects, and CRIOs can be the result of the composite/sequential execution of many different services. Clearly, our model can accommodate higher-order abstractions in scenarios, such as situation awareness applications, that call for the creation of actionable knowledge from the analysis of both raw data and processed information generated by a multitude of sensors and services. On the other hand, it is also possible that different user requests trigger the creation of different CRIOs starting from the same IOs.

To address the challenging communication issues of Smart City environments, CRIOs will be delivered as responses to users' requests through an Information-Centric Networking (ICN) inspired dissemination solution [28] [20]. ICN is a networking paradigm that moves the focus of communications from the source and destination to information itself. Thus, it can naturally take advantage of spontaneous networking opportunities to improve latency and delivery ratio in spite of disrupted communications, as well as seamlessly exploiting innovative functions provided by future cellular communications, such as LTE direct and 5G content caching [31]. However, while ICN solutions are typically based on content-centric networking, the SPF information dissemination adopts content-based publish/subscribe communication semantics that, by allowing nodes to express interests in any IO disseminated by a given service, is much better suited for dynamic networking and computing environments such as Smart Cities [53].

For an illustrative example, let us consider how the security check and logistics support applications illustrated in Section 3.0 could be implemented in the SPF service model. As depicted in Fig. 1, the first application would use two Fog services: a *1st order Security Control* one running on edge devices and a *2nd order Security Control* one running in the Cloud. The second application would use a *Crowd Monitoring* Fog service running on edge devices. In turn, those Fog services would be built on top of different sets of pipelines and services.

An *Anomaly recognition* pipeline running on an IoT gateway could process raw data consisting of image frames collected from video cameras in the device's proximity in the attempt to detect potentially hazardous situations, e.g., a man wielding a weapon. For each anomaly detected, the pipeline would generate an IO containing the anomalous portion of image. A *Face detection* pipeline could simultaneously process the same raw data frames and produce a set of IOs containing the portion of images that have been identified as a human face.

Both pipelines could send their output IOs to a *Security alert* service, which could further analyze and cross-correlate that information in order to reduce false positives, compare the faces detected against a reduced size database of local persons of interest, and generate geotagged security alert CRIOs to quickly alert police forces when necessary.

The Face detection pipeline could also send its output IOs to a *Logistics support* service that in turn could process that information, cross-correlating it with data on the number of smartphones currently active in the area, to generate CRIOs containing an updated estimate on the number of people in the area for the EMS teams' benefit. In addition, the same IOs could be forwarded to a *Police database comparison* service running in the Cloud that could compare the detected faces against a worldwide database of known subjects for a more accurate but slow search, and produce security alert CRIOs for police forces.

The *Police database comparison* service could also process the IOs produced by a *Face recognition* pipeline running in the Cloud and performing more computationally expensive analysis of raw data video feeds to produce IOs containing face identification points, which are much more effective to use for person of interest identification purposes.

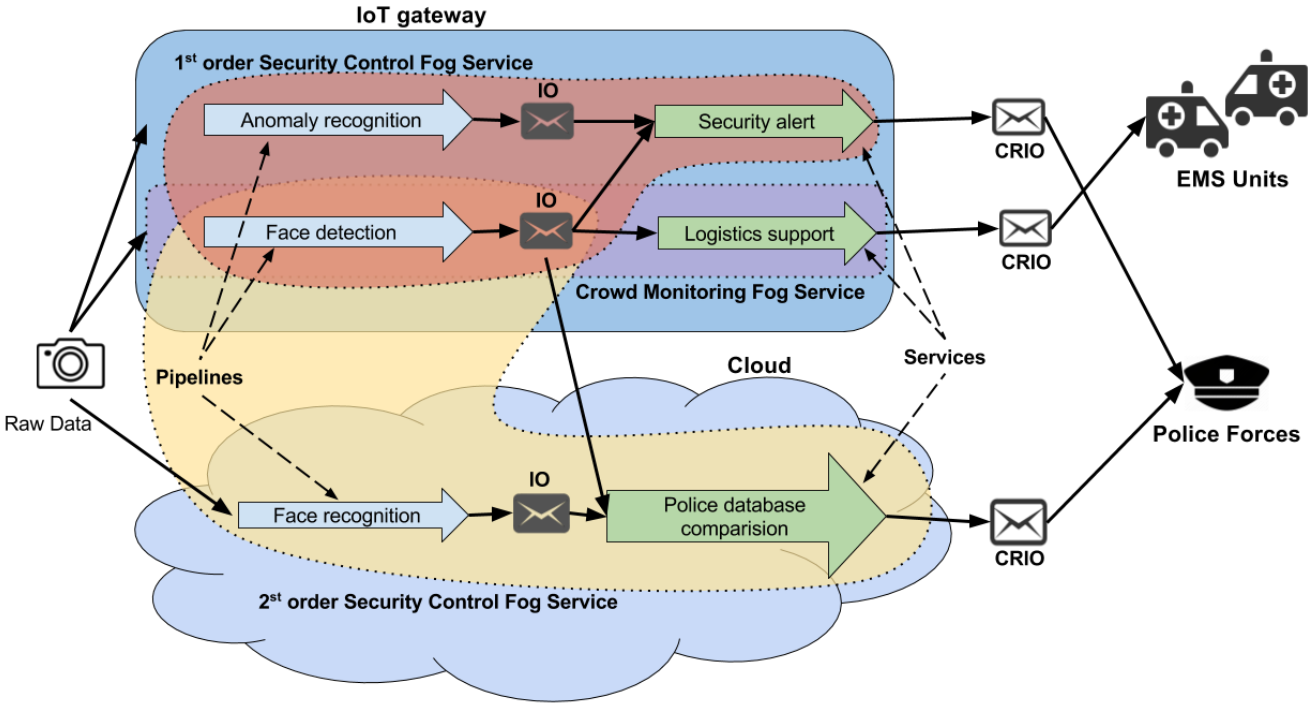


Fig. 1. The SPF information and service models.

3.2. Evaluating Value-of-Information for CRIOs

The SPF service model adopts techniques based on the Value-of-Information (VoI) concept to prioritize the processing and dissemination of important information [26]. VoI is an innovative concept that aims at estimating the usefulness of CRIOs according to their final recipient(s). More specifically, the CRIOs produced by a service are placed in a dissemination queue and ordered by their corresponding VoI. Additionally, the information dissemination strategy adopted by nodes promotes the dissemination of CRIOs with higher VoI values.

As it represents the level of interest that a user has on a specific CRIO, VoI is a dynamic value (typically decreasing over time) that does not depend only on the IO itself, but also on other factors concerning the user and the current context. Therefore, the VoI of a CRIO cannot be inferred only from its content: other contextual information is necessary for its computation.

VoI is calculated as a function of five factors:

$$VoI_{CRIO_n}(IO_n, fs, r, t) = SS(IO_n) * FSP(fs) * RN(r) * TRD(t, OT(r)) * PRD(OL(r), OL(IO_n))) \quad (1)$$

where IO_n is the IO from which $CRIO_n$ was built, fs is the Fog service responding to the users' requests, r is the (set of) recipient(s) for which IO_n can satisfy the requests, and t is the current time.

More specifically, $SS(IO_n)$ is a factor that allows to take into account service-specific considerations when assessing the value of the IO_n extracted from the data. $FSP(fs)$ is a factor that permits to assign higher VoIs to the CRIOs produced by higher priority Fog services. $RN(r)$ accounts for the number of requests that $CRIO_n$ can satisfy, normalized to the highest number of requests satisfied by a single CRIO within a recent time window. $TRD(t, OT(r))$, which stands for Timeliness Relevance (of Request) Decay, takes into account the time passed between request reception, provided by $OT(r)$, and response generation (the current time t). Finally, $PRD(OL(r), OL(CRIO_n)))$, for Proximity Relevance (of Request) Decay, is similar to $TRD(\cdot)$, but its impact on the final VoI score depends on the physical distance between the position of the requestor, given by $OL(r)$, and that of the source of IO_n, provided by $OL(IO_n)$ (in case r was simultaneously issued by several requestors, $OL(r)$ returns the location of the requestor closest to the IO source). For this purpose, service requests issued by applications will contain the current geographical position of the requestor node, e.g., obtained by means of a Global Positioning System (GPS).

Note that, while every service implementation has to provide its own *SS* function, the *FSP*, *RN*, *TRD*, and *PRD* functions are not service specific and can be provided directly by SPF. For instance, SPF currently offers two generic linear and exponential decay functions that services can leverage as *TRD* and *PRD* simply by setting the desired mode, i.e., linear or exponential, and the values for the corresponding control parameters. Developers will also need to assign a priority value to their Fog services, which SFP will automatically consider when calculating *FSP*.

For instance, both the security check Fog services in our running example would deal with very important CRIOs, having very high and slowly decaying VoIs. As a result, they would be configured with the highest priority and with control parameters that set a mild decay gradient for both the *TRD* and *PRD* functions. Instead, the Fog services used by the e-health application

would deal with very important but location-dependent CRIOs, as health information of citizens in other areas of the city would be the concern of other EMS teams. So, they would likely be configured with high priority (possibly lower than the one assigned to the security check Fog services) and with control parameters that set a mild decay gradient for *TRD* function and a high decay gradient for *PRD* function. Finally, most of the Fog services deployed in the context of commercial applications would deal with non-critical and time- and location-sensitive CRIOs, and thus be configured with relatively low priority and with control parameters that set a steep decay gradient for both the *TRD* and *PRD* functions.

4. The SPF Fog-as-a-Service Platform

SPF[1] is a Fog-as-a-Service platform that we developed in response to the peculiar needs and challenges raised by Smart City environments. SPF proposes a new information-centric and VoI-based service model that provides a powerful set of concepts and tools for the development and deployment of Fog applications. The SPF model allows the definition of self-adaptive and composition-friendly Fog services, which can be implemented on top of software components that can execute either on edge devices or in the Cloud, migrate to different computing platforms, and automatically scale their computation and bandwidth requirements according to the current execution context.

SPF provides a set of management concepts and tools to support the needs of the three groups of stakeholders that we envision will operate in Smart City scenarios: *platform providers*, *service providers*, and *end users*. Platform providers are the owners of a set (or platform) of Fog and/or Cloud resources that are publicly accessible, typically in a pay-per-use fashion. They are in charge of the administration and supervision aspects of the platform: installing, running, configuring, and maintaining SPF components and providing functions to federate with other platforms. Service providers develop IoT services, deploy them on Fog platforms, and take care of their initial configuration. Finally, end users access IoT services through client apps installed on their smart (phones or wearable) devices (service developers are also responsible for providing client applications for the users).

SPF has two main component types: Programmable IoT Gateways (PIGs) and Controllers. PIGs provide the functions of data processing and information dissemination in response to users' requests to the platform, and can be deployed either in the Fog (i.e., on edge devices) or in the Cloud. An instance of the PIG component must be deployed in each Fog or Cloud node that can run an SPF Fog service.

The Controller component allows application developers to define Fog applications that provide a set of Fog services and deploy Fog applications and Fog services on the PIGs accordingly. In addition, the Controller manages users' requests and forwards them to the relevant PIGs (i.e., those that have running instances of the corresponding Fog services) for processing. Finally, the Controller also implements sensor discovery functionalities and can connect PIGs to new data

[1] We released SPF as open source (MIT license) on the https://github.com/DSG-UniFE/spf Web page.

sources. By controlling service and application deployment, the Controller ensures that data processing on the PIGs occurs only when needed, such as in the presence of users' requests.

Fig. 2 presents an example deployment of SPF, which involves three different SPF platforms: two Fog-based ones (Platform A and Platform B) and a Cloud-based one (Platform C). The figure also displays how SPF processes a Fog service request issued by User 1 that involves the local Fog-based processing of locally generated raw data. First, the SPF App on User 1's mobile device sends the service request to the SPF Root Controller, that will typically be hosted in a Cloud environment. Service requests contain several information, including the service name, the timestamp of the request, the GPS position of the user issuing the request (used for VoI calculation purposes, as detailed in Section 3.2, and by the SPF Controller to select the closest Platform for request processing), service- and request-specific information (such as a query string), and so on. The SPF Root Controller then examines the request and forwards it to the Controller components managing the SPF platforms of interest: in the example case, Platform A. In turn, Platform A's Controller handles the request by activating (or reconfiguring, in case they were already active and the current configuration needs an update) service components in the PIG locations within the platform. In the example case, Platform A's Controller activates PIG P in proximity of WSN A, which will process the raw data collected from the WSN to produce CRIOs that will be disseminated to User 1 (for simplicity, Fig. 2 assumes that both the pipeline and the service components used to process raw data in response to User 1's Fog service request are activated on the same PIG).

SPF applications use traditional TCP/IP communications (depicted as dashed thin arrows in the Fig. 2) to send service requests to the SPF Controller. In turn, the CRIOs generated will be disseminated by an ICN-inspired solution that supports content-based publish/subscribe communications. This design choice enables SPF to tackle the communication issues that arise in highly dynamic mobile networks or when limited support from the network infrastructure is available, by leveraging ad hoc and device-to-device communication opportunities to implement multi-hop communications between nodes with no direct contact. More specifically, as detailed in Section 4.2, CRIOs are classified (or scoped, in ICN terminology) according to the Fog service(s) they were generated by and disseminated along service-specific "virtual channels". For instance, in Fig. 2 the CRIOs produced by the Fog service instantiated in Platform A follow a multi-hop dissemination path (depicted with thick yellow arrows), with User 2 acting as a relay between PIG P and User 1, which have no direct communications.

Note that other kinds of requests might instead involve the activation of service components in multiple SPF platforms. For instance, Fog services requiring computationally heavy, location independent, and/or time-insensitive processing might be performed in the Cloud-based Platform C.

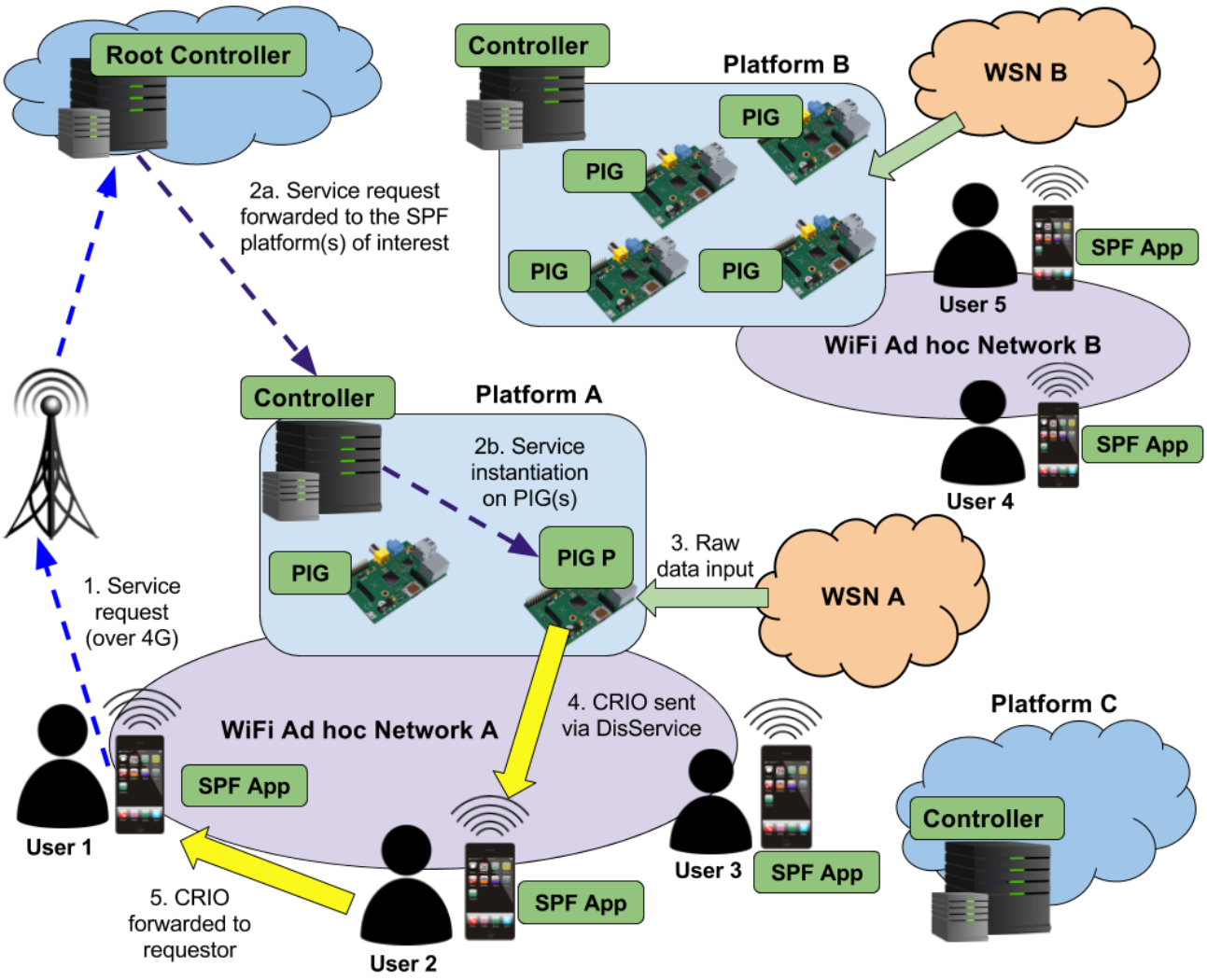


Fig. 2. An example of SPF deployment in a Smart City environment.

4.1. Service Activation and Deactivation

In SPF, Fog services are self-adaptive and can operate in two different modes according to the location of the corresponding PIG they run on top of: a "lossy" mode if the PIG is hosted on an edge device and a "full" mode if the PIG is hosted in the Cloud. When operating in lossy mode, SPF services automatically decrease their offered QoS, for example, by adopting less sophisticated analytics or analyzing only a part of the information they received. On the other hand, when operating in full mode, SPF services could run at the best QoS level, analyzing all available information with computationally expensive analytics and with the almost unlimited amount of resources available without strict latency requirements. In this way, SPF services seamlessly and autonomously adapt their resource demands for information processing to the current execution environment.

In addition, SPF allows the definition of two different types of Fog services: *on-demand* and *background*, with significantly different (idle) behavior and lifetimes. In fact, according to a continuous processing and information-centric paradigm, we envision that in large part Fog services will be normally off, and be executed on-demand only upon reception of one or more

corresponding user requests. A smaller number of Fog services will be always active, running continuously to provide mission-critical and/or background functions, such as those on which the security checks and e-health applications in our running example rely.

SPF is also in charge of the automatic de-instantiation of Fog services when they are not needed anymore. In this regard, the behavior of the corresponding software components (services and pipelines) depends on the type of Fog services registered with them. If at least one registered IoT service is a background service, the corresponding components will remain active and process incoming data continuously; otherwise, data processing is performed only if there is at least one pending request on the edge (or Cloud) device. This allows reducing resource (CPU, memory, and battery) consumption by deactivating unnecessary services and pipelines when possible.

Additionally, each background service has an associated maximum idle time, after whose expiration the IoT service (and the corresponding service components) will be automatically deactivated. If no value is assigned, the background service will run endlessly. Under certain circumstances, the platform might decide to change the configured idle lifetime for any Fog service running on an edge device, such as when many different Fog services are active, the remaining resources become scarce, or another application with higher priority is deployed.

4.2. CRIO Dissemination and Service Composition

Most ICN solutions implement content-centric networking, in which IO delivery is triggered by interest messages issued by content requestors. The networking infrastructure routes interest messages issued by consumers to the closest information holder, be it the source or another node that had previously cached the requested IO. Upon reception of an interest message that it can satisfy, an ICN node hands over the requested IO to the ICN infrastructure, which will forward the IO hop-by-hop to the requestor following the same path taken by the interest message in the opposite direction.

This communication semantic works well for the delivery of persistent IOs whose name (or id) is known by requestor – the main reference scenario that the ICN paradigm was designed to address. However, several studies discussing ICN adoption for mobile and ad hoc scenarios [52] argue that the content-centric networking communication model is not well suited for the delivery of transient IOs, which are significantly more relevant for Smart City applications leveraging real-time information sensing and processing [32]. The same studies indicate that content-based publish/subscribe communications would instead represent a much more effective solution in these scenarios [53].

In addition, content-based publish/subscribe communications do not require the ICN network to maintain up-to-date Forwarding Information Base (FIB) tables on all nodes across the network to enable the routing of interest messages. In a network with highly mobile nodes, frequent link creation and disruption, and a continuously changing topology, content-centric networking would force nodes to consume a large part of their resources just to refresh FIB entries. Similarly, the effectiveness of Pending Interest Tables (PIT) in routing IOs back to the requesters also suffers in highly dynamic scenarios, where nodes' mobility would often invalidate the paths from IO holders to the requestors established by interest messages.

To exploit the advantages provided by content-based publish/subscribe ICN approaches while avoiding the drawbacks of content-centric networking, CRIO dissemination in SPF relies on

DisService [28], an ICN-inspired dissemination middleware designed to provide efficient, thematic channel-based group communications in mobile ad hoc networks. DisService performs CRIO dissemination in an information-centric and content-based fashion, by assigning each CRIO a unique name and a service identifier (or scope, in ICN terminology) and disseminating it within a specific channel. DisService provides effective in-network caching capabilities by maintaining a cached CRIO database on each network node, a mechanism similar to the nodes' Content Store (CS) in traditional ICN solutions. The middleware also allows to define cache eviction policies at the single service level, that it consults to decide whether to discard obsolete CRIOs in case new or more relevant CRIOs are received. This allows SPF Controllers to configure service-specific cache eviction policies in DisService according to service semantics and users' preferences, a mechanism similar to the support for special name aliases such as "/sensor-id/temperature/latest" implemented by some ICN solutions.

DisService provides VoI-based prioritization to regulate the access to the network: CRIOs with higher VoI scores will be disseminated before responses with lower VoI scores. Additionally, during dissemination, DisService uses the VoI associated with each CRIO to regulate network resource sharing within each dissemination channel. The support for VoI-based information prioritization enables DisService to optimize critical tasks of ICN, namely forwarding and caching of the CRIOs produced by PIGs to promote the delivery and availability of most critical data to the interested recipients. Referring to the example in Fig. 1, this might include the delivery of CRIOs containing the approximate location of a person of interest that was spotted during a public event to police forces or of CRIOs that contain time-sensitive information that interest many users, such as the line-up of bands at a concert or the location of the least crowded food stands.

Together with in-network caching, which takes advantage of storage capacity on the network nodes, the ICN-inspired solution implemented by DisService enables effective CRIO delivery in the presence of mobile ad hoc networks and effectively supports group communications [20] [32]. The thematic channel-based group communications provided by DisService also represent the basic communication functions that SPF adopts to connect services with remote pipelines and to compose Fog services. By joining or leaving a DisService channel dedicated to the dissemination of the corresponding IOs or CRIOs, services and Fog services can connect with a pool of information sources and recipients – either services, Fog services, or users. For instance, in the example of Fig. 1, the IOs produced by the Face detection pipeline will be forwarded to the Security check and Logistics support services running on the same PIG and disseminated over a DisService channel subscribed by the Police database comparison service running on a remote PIG hosted in the Cloud.

In terms of network resource consumption, the SPF service and information dissemination models can tremendously reduce the amount of network traffic generated by processing raw data in the Fog. Additionally, the adoption of ICN-inspired content-based publish/subscribe solutions to convey responses to end users via spontaneous networking reduces the strain on the network infrastructure and allows applications to provide their services even when the infrastructure is overloaded, such as when too many people are connected to the same base stations (for example, during concerts or sport events). In fact, the dissemination mechanisms adopted by DisService have proved very effective in mobile and extremely dynamic networking scenarios, such as mobile ad-hoc and delay-tolerant networks [28].

4.3. Resource Management and Federation

SPF adopts several mechanisms to contain the consumption of memory and processing resources on edge devices. One solution is a direct consequence of the service model, which allows the reuse of pipelines. This system permits multiple services to register to the same processing pipeline; when this happens, the pipeline will deliver a copy of the produced IOs to each registered service. This avoids different applications performing the same operations on the input data, thus effectively allowing the reuse of processing resources and functions on the edge device and contributing to saving significant computational and memory resources under certain circumstances. For instance, the IOs produced by the Face detection pipeline illustrated in Fig. 1 are forwarded to 3 different services.

In addition, Fog services leverage content-based filtering techniques to reduce the volume of processed raw data by restricting processing to only those data that contain potentially novel information, in accordance with the preferences of application developers and platform managers. The rationale behind content-based filtering is to avoid processing raw data that do not differ significantly from those analyzed in the past, as they would probably not add a significant value to the information already available.

The amount of new content that a raw data message has to contain in order to pass the content-based filtering phase, and thus be selected for processing, can be easily controlled. As a result, content-based filtering is a very effective way of modulating the consumption of computational resources for Fog services, i.e., their "lossiness". To this purpose, each service has a threshold τ that dictates how different new data need to be, compared to the last processed data, to trigger data processing on the pipelines. A threshold of 0.0 entails the processing of all input data, while a value of 1.0 will trigger data processing only of the first raw data message received by each sensor. When multiple services register with the same pipeline, such as in the case of the Face detection pipeline mentioned above, the PIG considers all the Fog services using that pipeline and adopts the lowest value among the corresponding thresholds for content-based filtering purposes.

At the administrative level, SPF allows grouping resources in "platforms" that refer to a single provider and enables their federation, thus realizing a hierarchically distributed service management architecture. Each provider will deploy a Controller component that manages the resources available in its platform, and coordinates with a higher level Controller for service instantiation/de-instantiation and request dispatching. From the users' perspective, all the service requests will be directed to a root Controller component, which will coordinate with the Controllers of each registered platform to overview the execution of Fog services and activate/deactivate the corresponding software components (pipelines and services) according to the users' requests.

From the perspective of information source discovery, SPF implements both a spontaneous/decentralized discovery and configured/centralized directory approaches. In the first case, a discovery component running on the PIG performs sensor discovery in their proximity using multicast Domain Name System (mDNS) and DNS-based Service Discovery (DNS-SD). Discovery functions could be activated either in Controllers or in PIGs, respectively enabling providers to activate additional resources (e.g., a micro-Cloud) in a plug-and-play fashion and

enabling PIGs to detect the presence of sensors in their proximity and immediately start receiving and processing the related raw data.

On the other hand, providers could also manually add information sources and resources by configuring their Controller accordingly. This would be an effective way to integrate open data sources (street cameras, environmental monitoring systems like air pollution sensors, and so on) within SPF. The Controller will in turn notify the presence of those information sources to PIGs, which, in turn, will be able to connect with them.

5. Experimental Results

Fig. 3 illustrates the testbed scenario that we considered for the evaluation of SPF. To demonstrate the effectiveness of the SPF information and service model in enabling self-adaptive Fog services capable of automatically scaling their information processing functions to the particular hardware platform they run upon, we considered a simple but realistic scenario in which information processing tasks can be hosted either on edge and Cloud devices.

More specifically, the testbed consists of a root Controller, 2 different SPF platforms (a “Fog platform” in which PIGs are only hosted on edge devices and a “Cloud platform” in which PIGs are hosted on a public Cloud data center), and a Wireless Sensor Network (WSN) that ‘feeds’ raw data as a sequence of JPG images to the interested Fog services. The root Controller is hosted in the Amazon Web Services (AWS) Cloud computing platform, executing in a t2.micro VM instance with 1 virtual CPU and 1 GB of RAM, running AMI Linux. The Cloud platform is also hosted by AWS, and the relative Programmable IoT Gateways (PIGs) are executed in m4.2xlarge VM instances with 2.3 GHz Intel Xeon processors, 8 virtual CPUs and 32GB of RAM, running Ubuntu Linux 16.04.2 LTS. The Fog platform runs PIGs on top of Raspberry Pi 3 Single Board Computers with 1.2 GHz ARM processor and 1GB of RAM, running Linux Raspbian 8.0.

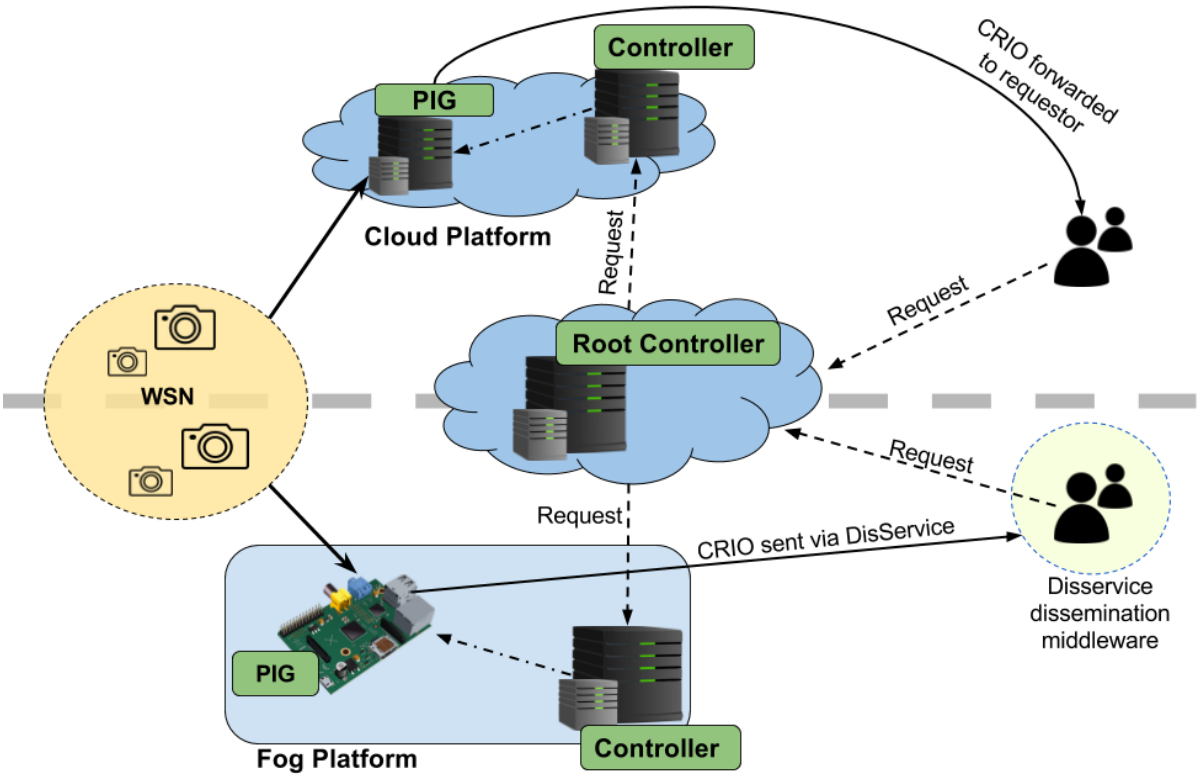


Fig. 3. Testbed architecture.

For the experiments, we developed a *Crowd monitoring* Fog service, modeled after the Fog service that provides the EMS teams an estimate of the number of people present in the Smart City center in our running example. The Crowd monitoring Fog service is built on top of a Face detection pipeline that takes as input images produced by cameras scattered in the city and feeds the generated IOs into a Face count service. To limit the processing latency, we implemented a mechanism that saves PIG resources. More specifically, the Face detection pipeline has an internal queue with a maximum capacity of 50 input raw data messages. When the pipeline receives a frame, it checks if the queue is full: in the affirmative case, it removes the 'oldest' packet from the queue; otherwise, it adds the data to the queue. This prevents the pipelines to incur the risk of filling the hard drive and/or the memory available on the node. Note that this strategy is in line with the information-centric and utility-based philosophy of the SPF service model, as removing the old elements and focusing the processing on the most recent raw data clearly represents a VoI-conserving policy.

The experiments have the objectives of evaluating the performance of the Crowd monitoring Fog service and understanding the tradeoffs and transition points between executing services on edge devices or in the Cloud. To this end, we deployed the Crowd monitoring Fog service alternatively on a PIG in the Cloud platform and on a PIG in the Fog platform, in both cases stressing the PIGs' capabilities to fully evaluate the relative advantages and disadvantages of the two configurations.

In order to provide the Fog service with reproducible input, we simulated the WSN sensors behavior with a Java application that always provides the same data set of raw images, of about 1.5MB in size each, during the different executions. We obtained those images by fragmenting a

video showing people walking in a city cross road. Finally, with respect to the pipeline processing, we implemented a simple face detection using the OpenCV open source library for computer vision.

5.1. Test 1: Varying Input Workload

In the first test we kept the content-filtering threshold constant to a value of $\tau = 0.03$, while varying the load of the input raw data. To do so, we simulated nine different data sources, with different data generation rates. More specifically, the sources can have 1, 2, or 4 different sensors, each one providing 250 consecutive images, and a time step between the generation of consecutive images of either 0, 500, or 1000 ms.

We executed the same tests on both platforms, measuring a set of significant parameters indicated with red arrows in Fig. 4. More specifically, we measured the number of raw data messages dropped because the incoming message queue filled up, the number of raw data messages discarded by the content-based filter, and the number of processed raw data messages. For data that pass the content-based filtering phase, i.e. data processed by pipelines, we measured the relative queue time into the PIG and the CPU time for the processing.

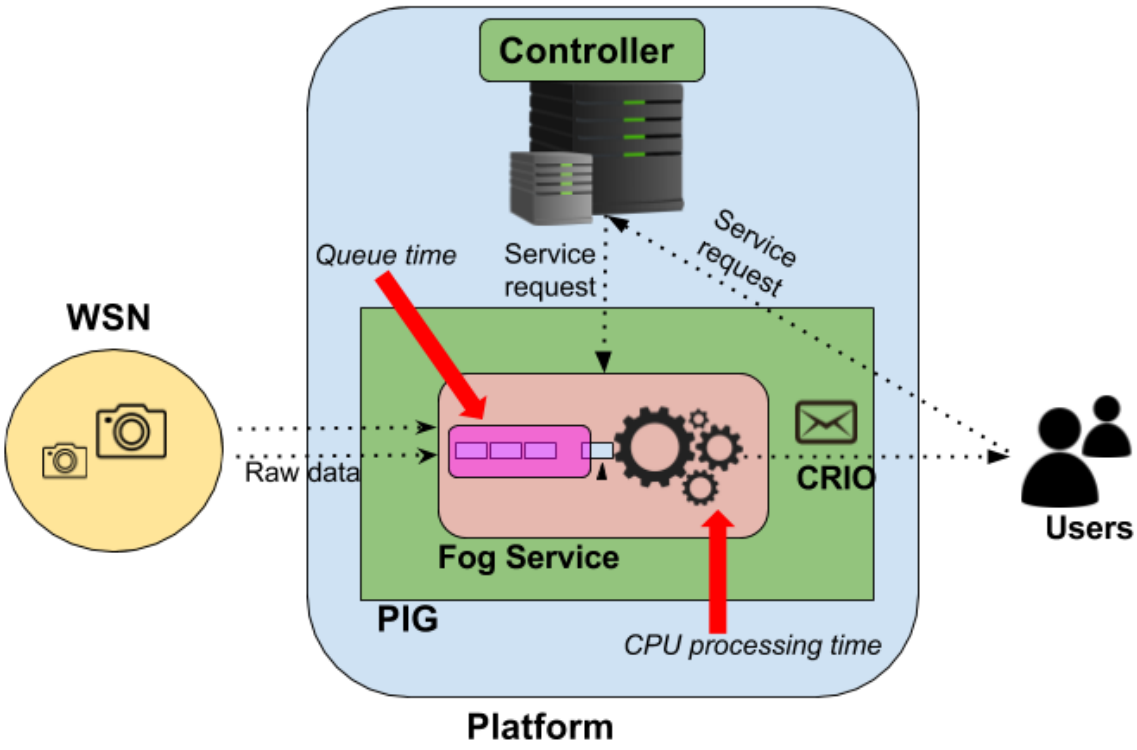


Fig. 4. Measured parameters for each SPF platform.

Fig. 5 and Fig. 6 plot the results we obtained by running the Crowd monitoring Fog service on top of a PIG respectively in the Fog platform and in the Cloud platform, for each of the 9 different input source configurations. To focus on the most significant results that characterize a real scenario, we

show only the trends that, in temporal order, are included from the beginning of the execution to the point when a steady-state condition is reached.

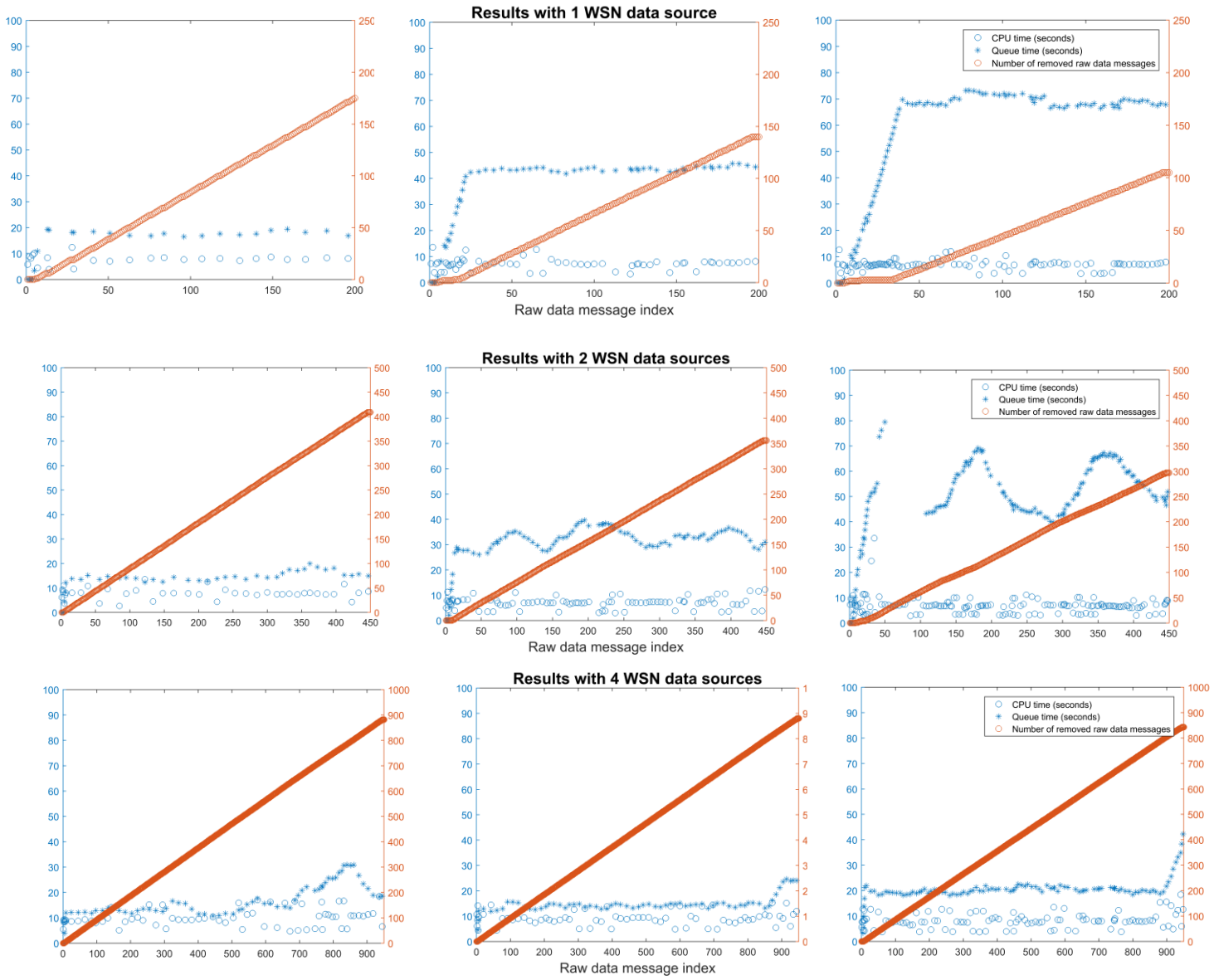
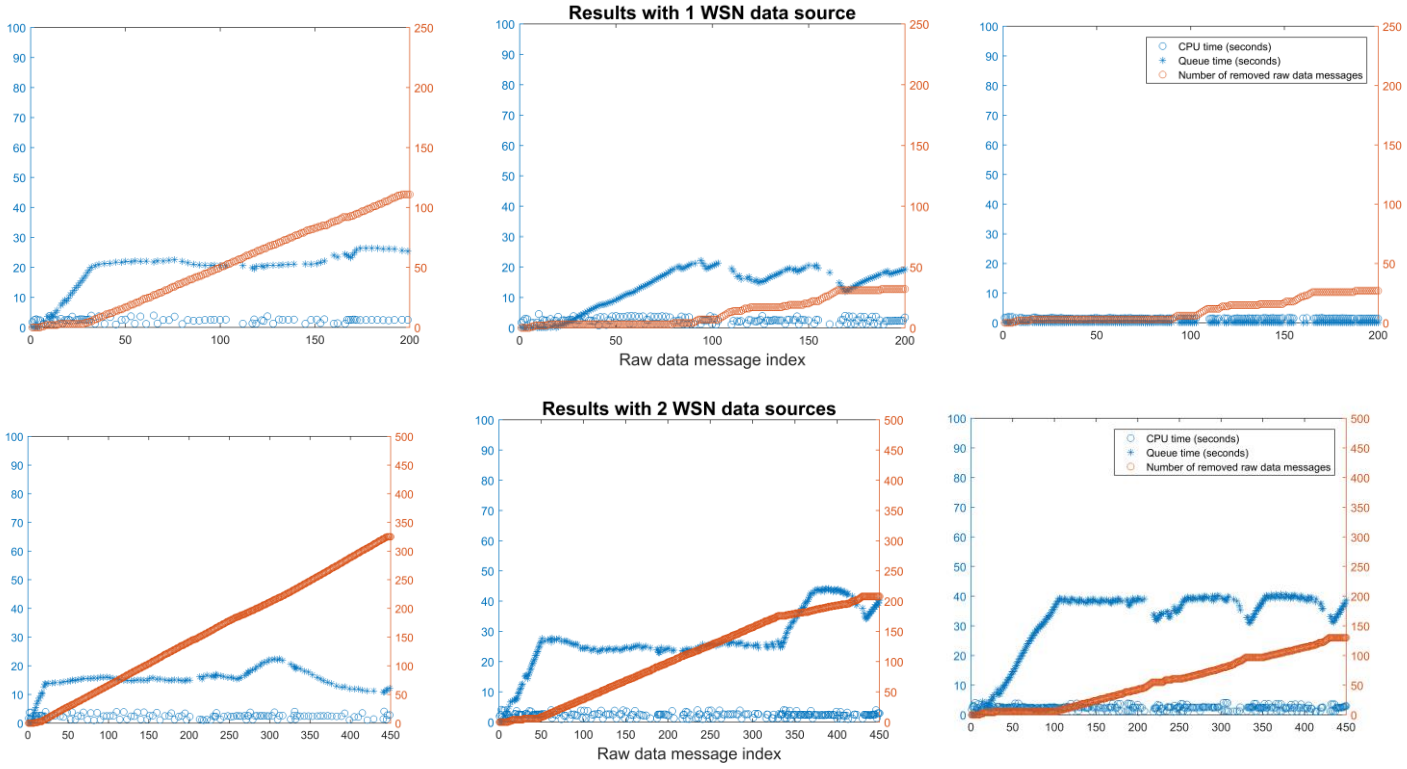


Fig. 5. Results of the Fog platform with variable load. The graphs in the first, second, and third columns respectively depict the results obtained for a time step of 0ms, 500ms, and 1000ms.

As shown in Fig. 5, the trend of the various parameters reflects the behavior that we expected: for each given number of WSN sensors (i.e. for each row of graphs in the figure), as the time step increases (i.e., the data rate decreases) we see a lower number of dropped packets but a higher queuing times for the processed data. This is due to the fact that the packets are dropped from the queue with a lower frequency, so the items that will be processed wait for longer times. With four WSN sensors (the last row of graphs in the figure) this trend is less evident and the number of removed raw data messages is substantially independent from the time step. This is due to the fact that the data received by four sources overloads the limited computational resources of the PIG in the Fog platform and cause congestion at the queue level.



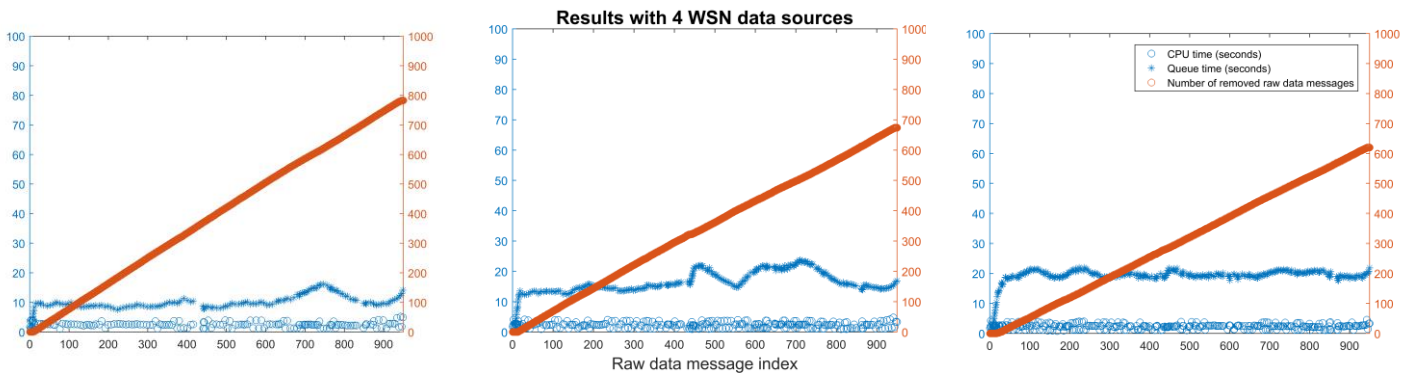


Fig. 6. Results of the Cloud platform with variable load. The graphs in the first, second, and third columns respectively depict the results obtained for a time step of 0ms, 500ms, and 1000ms.

Unsurprisingly, Fig. 6 shows that the Cloud platform has significantly better capabilities that the Fog one. We see a general decrease of CPU times and a greater density of the elements, which means more processed data and faster processing speed. However, note that the queuing times observed in the Fog platform are comparable with those of the Cloud one, especially for the 4 WSN sources case which represents the most computationally intensive one. This demonstrates that SPF's information and service models are very effective in enabling the development of Fog services that can guarantee time-sensitive information processing by seamlessly discarding some information to adapt to the resource constraints of the devices in which they are hosted.

In addition, while the PIG hosted in the Cloud platform is capable of processing almost the totality of the input raw data with one WSN sensor (first row of graphs in the figure), the amount of raw data to process in the configurations with two and four WSN sensors (second and third rows of graphs in the figure) is enough to overload the non-negligible capabilities of a Cloud-hosted PIG as well. This demonstrates that exploring trade-offs between processing speed and accuracy for information processing task can bring advantages not only for Fog services hosted in edge devices, but for those hosted in the Cloud as well.

5.2. Test 2: Varying Content-filtering Threshold

The strategy for the second test is to keep constant the input workload, produced by two WSN sensors with a time step of 0 milliseconds, and varying instead the content-filtering threshold parameter, with values ranging from 0.00 to 0.25. Fig. 7a and Fig. 7b show the means and variances of the CPU and queue time obtained, respectively, for the Fog and the Cloud SPF platform.

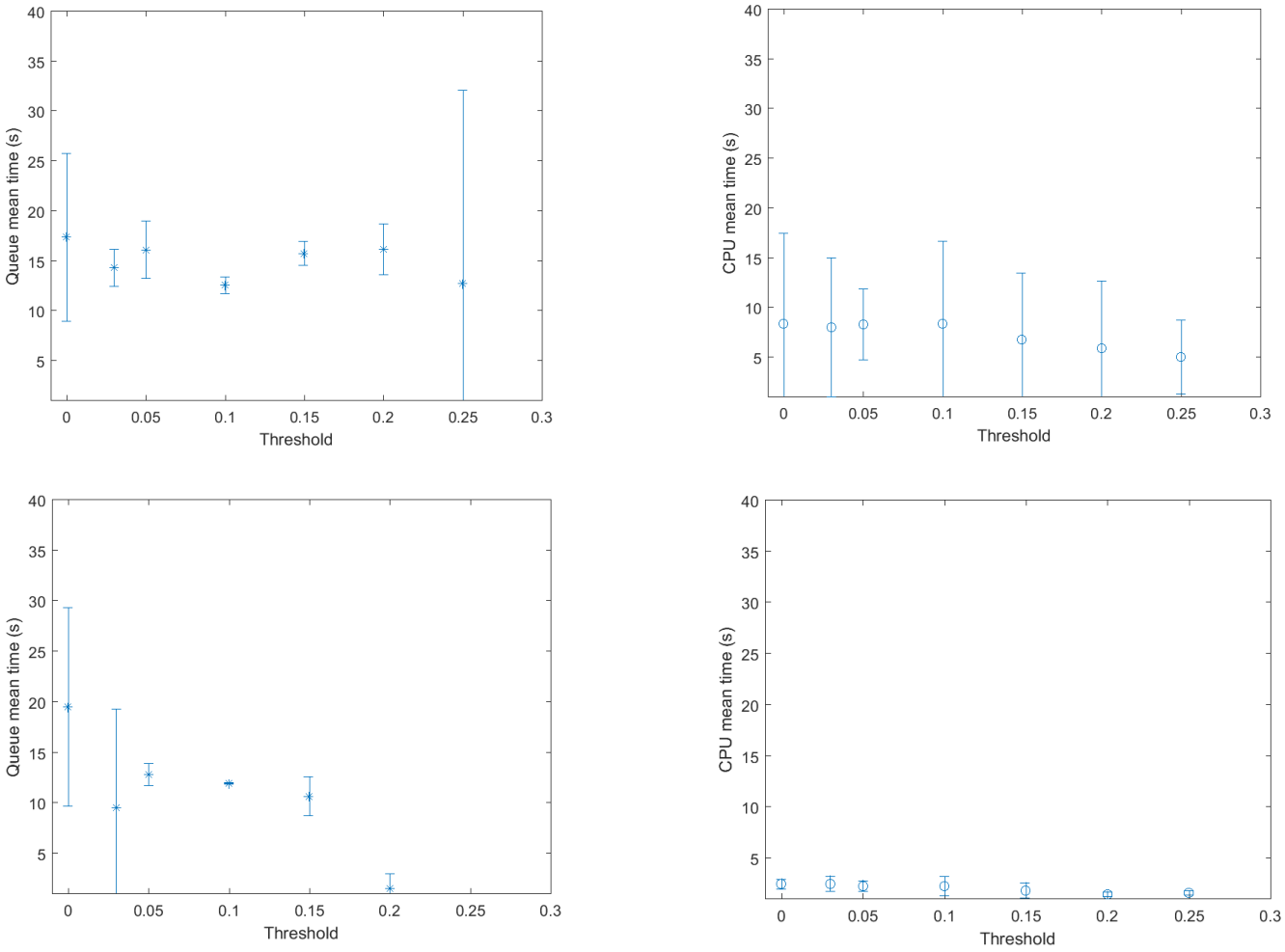


Fig. 7. CPU mean time and queue mean time with constant input workload and variable content-based filtering threshold. The results for the Fog platform are shown in the first row, those for the Cloud platform on the second row.

The results show a considerable performance difference between the Fog and the Cloud platform, especially in terms of CPU time, as expected. However, a significantly more interesting aspect to note is that the content-filtering phase is capable of containing CPU and queue times in a very effective way even in the resource-constrained PIG hosted in the Fog platform.

Fig. 8 and 5.7 compare the number of raw data messages dropped from the queue (red), of raw data messages actually processed by pipelines (blue) and of raw data messages discarded by the content-based filtering phase (yellow).

In general, the number of messages dropped from the queue is lower in the Cloud platform for each threshold. However, for very low (0.05 or less) threshold values, the difference is lower than one might expect. If we factor in the communication latencies involved, in those cases it could be convenient to execute the processing on the Fog platform, despite the loss of more packets. Instead, for threshold values in the range [0.10 - 0.25] the number of dropped packets decreases drastically in the Cloud.

These results support once again the claim that, for information processing tasks, edge devices represent a valid alternative to Cloud in many cases, and that the lossy service model implemented by SPF presents interesting advantages not only for Fog services hosted on edge devices but for those hosted in Cloud as well.

Finally, these results allow to draw some generally applicable conclusions about the development and deployment of Fog services. More specifically, they suggest considering the tradeoffs in Fog vs Cloud location of service components at design time, by defining VoI calculation factors that are suited for the Fog service semantics as well as a proper content-filtering threshold.

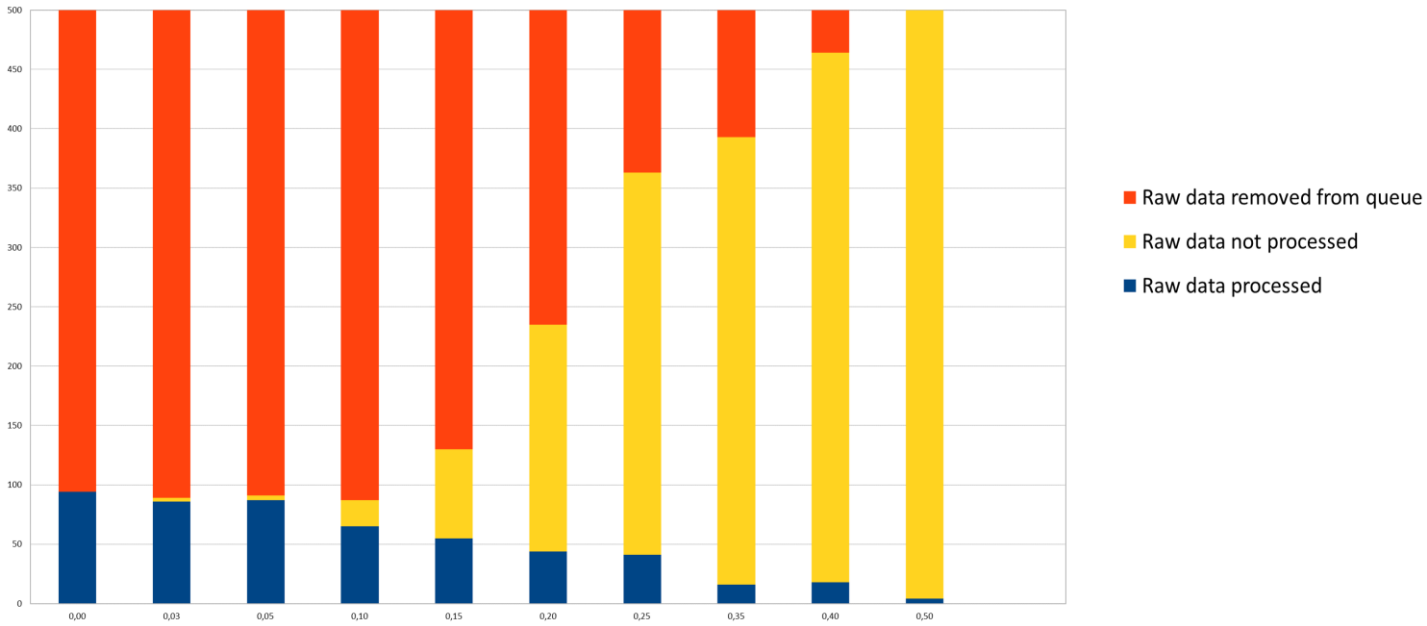


Fig. 8. Report on removed/processed/not processed input raw data, on Fog platform.

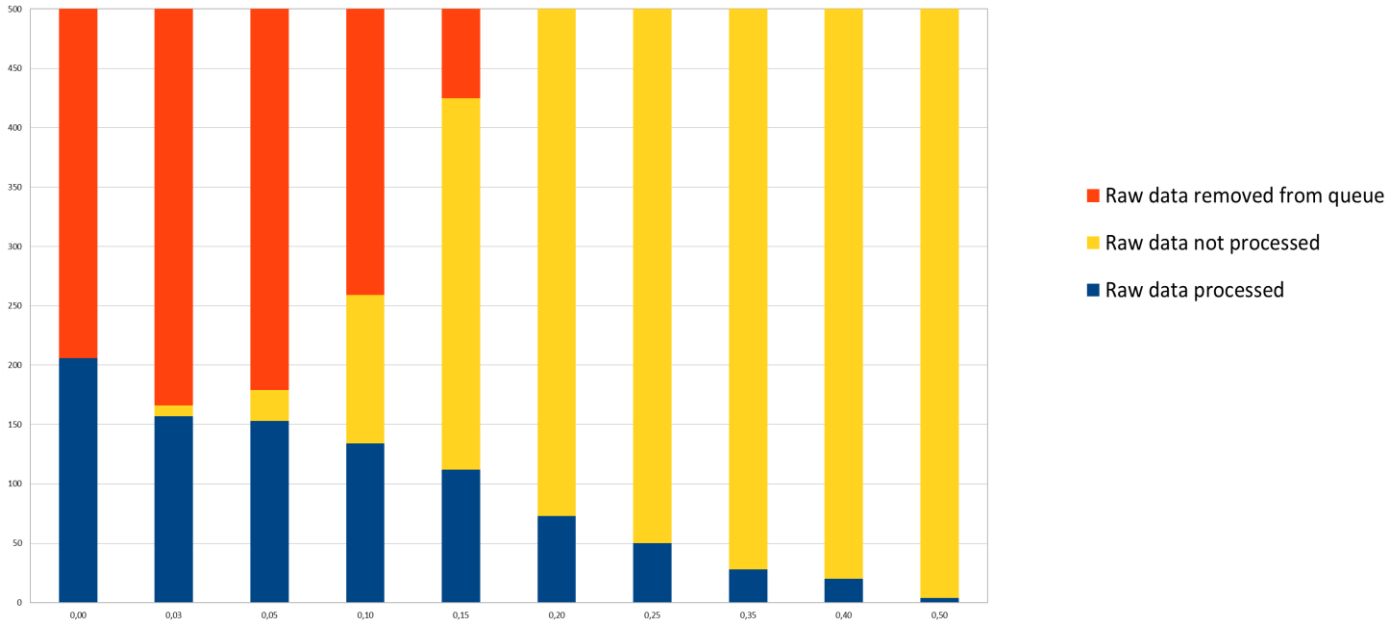


Fig. 9. Removed/processed/not processed input data, on Cloud platform.

6. Literature Review

While Fog Computing is still a recent paradigm, it has received a considerable attention from scientific literature. In this Section, we provide a survey of the research contributions that, like the work presented in this paper, focus on the architectural and programming model aspects in Fog Computing solutions.

6.1. Architectural Visions

Three different visions emerge from studies on Fog Computing. Researchers adopting a Cloud-centric perspective tend to perceive and present Fog Computing as an extension of the Cloud [10]. Under such vision, the Fog becomes an edge cloud approach that brings computation and storage resources closer to the consumer nodes and data sources [33] [34]. This enables highly responsive Cloud services, reducing ingress bandwidth into the Cloud through edge data analytics and preprocessing, and masking temporary Cloud outages [35].

This kind of approach typically relies on virtualization to manage processing and storage resources at the edge, as well as to package, deploy, and orchestrate applications running on the edge nodes [36], and to provide the necessary network functionalities and abstractions to ensure data transfer between applications and nodes. From this point of view, proposals based on software-defined networking (SDN) have emerged [37].

On the other hand, researchers with an IoT-centric perspective tend to present Fog Computing as a service platform rather independent and often disconnected from the Cloud [19]. Under this vision, Fog Computing platforms implement a distributed system that does not rely on the Cloud to provide the necessary processing power, storage, or service orchestration [38].

Research efforts classified in this area focus on the efficient and dynamic deploy of applications across the Fog nodes, e.g., using Kubernetes to allocate containerized applications on the nodes [39]. The optimization of task scheduling is another fundamental aspect of IoT-centric visions of Fog Computing, a process that needs to take into account applications' quality of service (QoS) requirements and variables such as nodes location, network latency, processing power available, and application characteristics [40].

Finally, other researchers highlight the potential benefits coming from the interaction of Fog and Cloud in a mutually supporting fashion [41] [42]. The idea is to create a service continuum between the Fog and the Cloud, potentially leading to a Fog-as-a-Service model [14], where Fog nodes can leverage the Cloud, e.g., to offload computation intensive tasks or store historical data, and the Cloud can also exploit the Fog, e.g., to support time-critical services with lower computational requirements [43]. Solutions in this area of Fog Computing include platforms based on smart gateways to deploy and connect multiple applications at the edge, enable application migration between different gateways, and provide communication and integration capabilities with the Cloud [44].

Research challenges involve both vertical integration, to provide device-Fog, device-Cloud, and Fog-Cloud communications mechanisms that are transparent to the applications, and horizontal integration, which deals with heterogeneous applications and systems and provides inter-platform and cross-communication support so that resources from assorted devices can interact. In addition, the interplay between Fog and Cloud that arises from this vision calls for solutions to optimize task assignment and relocation from the Fog to the Cloud and vice-versa, in order to reduce costs and improve applications QoS [45].

SPF naturally belongs to this last class of solutions. While Cloud-centric approaches aim at reusing the same technologies and deployment platforms available on the Cloud at the edge of the network, they will suffer from performance and coordination issues due to the heavy use of network, system, and resource virtualization in a severely constrained and distributed environment. At the other end of the spectrum, IoT-based approaches to Fog Computing tend to present a very fragmented view of resources to the applications; in addition, they often do not rely on support from the Cloud, which make the processing of intense computational tasks a challenge. By exploiting both the Fog and the Cloud, distributing computation tasks between central and edge resources depending on their load, defining a common paradigm to provide applications with access to those resources, and supporting service and resource discovery, SPF tries to respond to many open research challenges in the area and provide an effective platform for Fog-enabled application development.

6.2. Programming models

In Cloud computing, Web services (usually adopting 3-tier architectures and delivered using either IaaS, PaaS, or SaaS models) have emerged as a standard, and well-known players such as Amazon provide customers with enterprise class platforms and a plethora of sophisticated and robust services. Fog Computing is not quite there yet. While the Fog Computing paradigm is being intensively investigated, there is no commonly adopted standard for programming models and service architecture, either de jure or de facto.

Cloud computing programming models have been proposed for Fog Computing as well. Many research efforts focus on virtualization to manage resources on the Cloud and Fog nodes and to deploy applications [46] [33] [37] [36] [39]. OpenVolcano is a platform that exploits Network Function Virtualization (NFV) and SDN to provide scalable and sustainable smart services to end users. OpenVolcano implements an extension of the IaaS model for Fog Computing that can move computational and storage resources from the Cloud and user devices to the edge network [46]. Differently, Pahl et al. propose a platform to provide a PaaS environment at the network edge on clusters of single board PCs like the Raspberry Pi [33]. Their platform enables application deployment using Docker as the containerization technology and, in their study, they suggest using TOSCA (Topology and Orchestration Specification for Cloud Applications) to handle orchestration. [37] is another study that proposes the use of Docker containers to deploy applications on the network switches at the edge. Their framework leverages SDN concepts to automatically download, install, run, stop, and uninstall containerized applications on the network switches. Hong et al. also lean toward the use of containerized applications, e.g., Linux containers (LXC) or Docker, and a management system such as Kubernetes, OpenStack, or SaltStack [39] to deploy applications at the edge network.

However, the IaaS programming models, based on VM activation and migration, is too computationally expensive for Fog Computing applications [19]. In addition, while the recent development of OS-based virtualization technologies, pioneered by FreeBSD jails and Solaris zones and later exploded with LCX, produced mature and comprehensive solutions like Docker, which allow the packaging and deployment of applications with unprecedentedly low overhead levels, these tools are still too computationally expensive for Fog Computing environments.

In accordance with these observations, some researchers have proposed different approaches. Stack4Things is a Fog Computing platform for IoT applications that defines a Cyber-Physical System with Functions Virtualization (CPSFV) to manage smart objects (sensors and actuators), group them together, allow their mutual interaction, and enable the specialization of their behavior [21]. The last task is performed via contextualization, defined as the Cloud-controlled injection of code, in the form of plugins, into any smart object managed by the Stack4Things platform. Manzalini and Crespi [47] propose the Edge Operating System (EOS), a software architecture that leverages concepts and tools from SDN and NFV to exploit the processing power of network infrastructure elements. EOS is based on the Robot Operating System (ROS, http://www.ros.org), which defines the concept of nodes (processes that can potentially execute on different hosts), services that can be invoked by means of message exchange, and supports publish-subscribe communications between ROS nodes. EOS extends ROS mainly by adding the support for orchestrating task execution requests among EOS nodes capable of serving them, taking into account traffic congestion problems that might arise from un-optimized task allocations.

At the data analytics level, the stream programming model, which has been popularized by Big Data analytics in Cloud Computing environments, is rather limited. Apache Storm, one of the most interesting Cloud based solutions for analytics, enables to define services as information sources (nuts) and processing points (bolts). However, the model does not provide any automatic way to scale-down service processing if, e.g., a bolt is run in an edge device. Furthermore, Storm does not consider the problem of disrupted communications in Fog environments.

Giang et al. [48] present a Distributed Dataflow (DDF) programming model for the IoT that can take advantage of computing resources across the Fog and the Cloud. DDF enables the deploying of flows over multiple physical devices, which can process one or more nodes of the flow (namely, sub-flows). To ease application development, the authors extended Node-RED (NR, https://nodered.org), a visual flow-based programming language and runtime, to support the design of dataflows that run across edge devices and between the Fog and the Cloud. They named this new tool Distributed Node-RED (D-NR). Nodes use MQTT to exchange data and keep track of the status of the flows. Fog Computing Internet of Things (FC-IoT) is a paradigm that aims at improving the efficiency of solutions that leverage the Fog to perform tasks that are too computationally heavy to be carried out by smart objects [49]. The authors present a platform that manages resources organized into three tiers: IoT smart objects, Fog Computing, and Cloud Computing. Their platform provides communication capabilities to enable smart objects to communicate among each other, as well as to offload data and computation tasks to the upper layers (communication layer). In addition, the FC-IoT platform enables both runtime parameter reconfiguration and runtime code reconfiguration (that is, the remote update of the running code) on the Fog and Cloud nodes (computing layer). More specifically, code reconfiguration is made possible by the use of REEL, a lightweight VM specifically designed for data stream processing in IoT scenarios [50]. An intelligent processing layer provides energy management, input data quality assessment, and application adaptation to the dynamic environment.

Compared with these solutions, that were designed to process the entire set of raw data available and/or leverage traditional TCP/IP communications, SPF represents a comprehensive solution that proposes information and service models specifically designed to favor service composition and to prioritize important information for processing and dissemination. SPF enables to realize Fog services that achieve effective trade-offs between information processing speed and accuracy by adopting user-centric and value-based criteria and by building on top of components which are naturally and seamlessly capable of adapting their resource consumption to their current execution context.

Finally, let us note that the set of abstractions for development of IoT applications has just started being investigated [51] and that we expect research to increasingly focus on these aspects in the near future.

7. Conclusions

To address the challenges of the Smart City environments, Fog Computing solutions need to explore the opportunities involved in deploying a portion (or even the entirety) of the information processing tasks traditionally executed in Cloud data centers directly on *edge devices*, (re)configuring the task allocation in dynamical fashion according to the current environmental and operating conditions and service objectives.

This paper demonstrated that the adoption of purposely designed information and service models presents significant advantages for the realization of self-adaptive and composition-friendly Fog services. More specifically, the information-centric and value-based service model proposed by our SPF Fog-as-a-Service platform enables the streamlined development and management of Fog services that can be either in edge devices, i.e., in the Fog, or in the Cloud, migrate to different computing platforms, and automatically scale their computation and

bandwidth requirements according to the current execution context. The approaches adopted by SPF to explore interesting trade-offs between information processing speed and accuracy, leveraging Value-of-Information based prioritization and content-based filtering, have proved to be very effective and capable of bringing important advantages not only for Fog services running on edge devices, but also for those running in the Cloud.

References

[1] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, “Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach”, Future Generation Computer Systems, February 2017.

[2] B. Rashid, M. H. Rehmani, “Applications of wireless sensor networks for urban areas: A survey”, Journal of Network and Computer Applications, Vol. 60, pp. 192-219, January 2016.

[3] Y. Saleem, N. Crespi, M. H. Rehmani, R. Copeland, D. Hussein, E. Bertin, "Exploitation of social IoT for Recommendation Services", IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, pp. 359-364, 2016.

[4] M. Govoni et al., "Enabling Social- and Location-Aware IoT Applications in Smart Cities", in Proceedings of 2nd EAI International Conference on Smart Objects and Technologies for Social Good, November 30-December 1, 2016, Venice, Italy.

[5] H. Amer, N. Salman, M. Hawes, M. Chaqfeh, L. Mihaylova, M. Mayfield, “An Improved Simulated Annealing Technique for Enhanced Mobility in Smart Cities”, Sensors Journal, Vol. 16, No. 7, June 2016.

[6] J. Liu, J. Li, L. Zhang, F. Dai, Y. Zhang, X. Meng, J. Shen, “Secure intelligent traffic light control using fog computing”, Future Generation Computer Systems, February 2017.

[7] A. Botta, W. De Donato, V. Persico, A. Pescapé, “Integration of Cloud computing and Internet of Things: A survey”, Future Generation Computer Systems, Vol. 56, pp. 684-700, March 2016.

[8] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions”, Future Generation Computer Systems, Vol. 29, No. 7, pp. 1645-1660, September 2013.

[9] A. Papageorgiou, B. Cheng, E. Kovacs, “Real-Time Data Reduction at the Network Edge of Internet-of-Things Systems”, in Proceedings of 11th International Conference on Network and Service Management (CNSM), 2015.

[10] A.V. Dastjerdi, R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential", IEEE Computer, Vol. 49, No. 8, August 2016.

[11] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog computing and its role in the internet of things", in Proceedings of the first edition of the MCC workshop on Mobile cloud computing (MCC '12), ACM, New York, NY, USA, 13-16.

[12] N. Fernando, S. W. Loke, W. Rahayu, “Mobile cloud computing: A survey, Future Generation Computer Systems”, Vol. 29, No. 1, pp. 84-106, January 2013.

[13] E. Ahmed, M.H. Rehmani, "Mobile Edge Computing: Opportunities, solutions, and challenges", Future Generation Computer Systems, Vol. 70, pp- 59-63, May 2017.

[14] M. Chiang, T. Zhang, "Fog and IoT: An Overview of Research Opportunities", IEEE Internet of Things Journal, Vol. 3, No. 6, pp. 854-864, December 2016.

[15] L.M. Vaquero, L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of Fog Computing", SIGCOMM Computer Communication Review, Vol. 44, No. 5, pp. 27-32, ACM, October 2014.

[16] R. Roman, J. Lopez, M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges", Future Generation Computer Systems, November 2016.

[17] Cisco Systems Inc., "Cisco Global Cloud Index: Forecast and Methodology, 2015–2020", 2016, available at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html (retrieved on April 7th, 2017).

[18] M. Zahran, "Heterogeneous Computing: Here to Stay", Communications of the ACM, March 2017.

[19] Y. Elkhatib et al., "On using Micro-Clouds to Deliver the Fog", Internet Computing, In press.

[20] A. Morelli, M. Tortonesi, C. Stefanelli, N. Suri, "Information-Centric Networking in next-generation communications scenarios", Journal of Network and Computer Applications (JNCA), Vol. 80, pp. 232-250, February 2017.

[21] D. Bruneo, S. Distefano, F. Longo, G. Merlino, A. Puliafito, V. D'Amico, M. Sapienza, G. Torrisi, "Stack4Things as a fog computing platform for Smart City applications", IEEE Conference on Computer Communications Workshops (INFOCOM 2016), San Francisco, CA, pp. 848-853, 2016.

[22] F. Petroni, L. Querzoni, R. Beraldi, M. Paolucci, "Exploiting user feedback for online filtering in event-based systems", Future Generation Computer Systems, Vol. 71, pp. 202-211, June 2017.

[23] R. Howard, "Information value theory", IEEE Transactions on Systems Science and Cybernetics, Vol. 2, No. 1, pp. 22-26, 1966.

[24] S. Galanis, "The value of information under unawareness", Journal of Economic Theory, Vol. 157, pp. 384–396, 2015.

[25] J. Quiqqin, "The value of information and the value of awareness", Theory and Decision, Vol. 80, No. 2, pp. 167–185, 2015.

[26] N. Suri et al., "Exploring Value of Information-based Approaches to Support Effective Communications in Tactical Networks", IEEE Communications Magazine, Vol. 53, No. 10 (Special Feature on Military Communications), pp. 39-45, October 2015.

[27] L. Bölöni, D. Turgut, "Value of information based scheduling of cloud computing resources", Future Generation Computer Systems, Vol. 71, pp. 212-220, June 2017.

[28] G. Benincasa, A. Morelli, C. Stefanelli, N. Suri, M. Tortonesi, "Agile Communication Middleware for Next-generation Mobile Heterogeneous Networks", IEEE Software, ISSN 0740-7459, Vol. 31, No. 2 (Special Issue on Next Generation Mobile Computing), pp. 54-61, March-April 2014.

[29] A. Bougueattaya et al., "A Service Computing Manifesto: The next 10 Years", Communications of the ACM, April 2017.

[30] T. Nakada, H. Nakamura (Eds.), "Normally-Off Computing", Springer, 2017.

[31] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, M. Varvello. “From content delivery today to information centric networking”, Computer Networks: The International Journal of Computer and Telecommunications Networking, Vol. 57, No. 16, pp. 3116-3127, November 2013.

[32] M. Amadeo, C. Campolo, A. Molinaro, G. Ruggeri, “Content-centric wireless networking: A survey”, Computer Networks, Vol. 72, pp. 1-13, October 2014.

[33] C. Pahl, S. Helmer, L. Miori, J. Sanin, B. Lee, "A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters", IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW 2016), Vienna, pp. 117-124, 2016.

[34] M. Aazam, E. N. Huh, "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm", in IEEE Potentials, Vol. 35, No. 3, pp. 40-44, May-June 2016.

[35] M. Satyanarayanan, "The Emergence of Edge Computing", Computer, Vol. 50, No. 1, pp. 30-39, January 2017.

[36] M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, "Osmotic Computing: A New Paradigm for Edge/Cloud Integration", IEEE Cloud Computing, Vol. 3, No. 6, pp. 76-83, Nov.-Dec. 2016.

[37] Y. Xu, V. Mahendran, S. Radhakrishnan, "SDN docker: Enabling application auto-docking/undocking in edge switch", IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS 2016), San Francisco, CA, pp. 864-869, 2016.

[38] D. Ye, M. Wu, S. Tang and R. Yu, "Scalable Fog Computing with Service Offloading in Bus Networks", IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud 2016), Beijing, pp. 247-251, 2016.

[39] H. J. Hong, P. H. Tsai, C. H. Hsu, "Dynamic module deployment in a fog computing platform", 18th Asia-Pacific Network Operations and Management Symposium (APNOMS 2016), Kanazawa, pp. 1-6, 2016.

[40] V. Cardellini, V. Grassi, F. L. Presti, M. Nardelli, "On QoS-aware scheduling of data stream applications over fog computing infrastructures”, IEEE Symposium on Computers and Communication (ISCC 2015), Larnaca, Cyprus, pp. 271-276, 2015.

[41] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing", IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Athens, pp. 325-329, 2014.

[42] W. Li, I. Santos, F. C. Delicato, P. F. Pires, L. Pirmez, W. Wei, H. Song, A. Zomaya, S. Khan, “System modelling and performance evaluation of a three-tier Cloud of Things”, Future Generation Computer Systems, Volume 70, pp. 104-125, May 2017.

[43] M. G. R. Alam, Y. K. Tun, C. S. Hong, "Multi-agent and reinforcement learning based code offloading in mobile fog", International Conference on Information Networking (ICOIN 2016), Kota Kinabalu, pp. 285-290, 2016.

[44] N. Verba, K.-M. Chao, A. James, D. Goldsmith, X. Fei, S.-D. Stan “Platform as a service gateway for the Fog of Things”, Advanced Engineering Informatics, November 2016.

[45] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, A. Y. Zomaya, "Cost-effective processing for Delay-sensitive applications in Cloud of Things systems", IEEE 15th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, pp. 162-169, 2016.

[46] R. Bruschi, P. Lago, G. Lamanna, C. Lombardo, S. Mangialardi, "OpenVolcano: An Open-Source Software Platform for Fog Computing", 28th International Teletraffic Congress (ITC 28), Würzburg, Germany, pp. 22-27, 2016.

[47] A. Manzalini, N. Crespi, "An edge operating system enabling anything-as -a-service", IEEE Communications Magazine, Vol. 54, No. 3, pp. 62-67, March 2016.

[48] N. K. Giang, M. Blackstock, R. Lea, V. C. M. Leung, "Developing IoT applications in the Fog: A Distributed Dataflow approach", 5th International Conference on the Internet of Things (IOT), Seoul, pp. 155-162, 2015.

[49] C. Alippi, R. Fantacci, D. Marabissi, M. Roveri, "A Cloud to the Ground: The New Frontier of Intelligent and Autonomous Networks of Things", in IEEE Communications Magazine, Vol. 54, No. 12, pp. 14-20, December 2016.

[50] C. Alippi, R. Camplani, M. Roveri, L. Vaccaro, "REEL: A real-time, computationally-efficient, reprogrammable framework for Wireless Sensor Networks", Proceedings of IEEE SENSORS 2011, Limerick, pp. 1193-1196, 2011.

[51] F. Zambonelli, "Key Abstractions for IoT-Oriented Software Engineering", IEEE Software, Vol. 34, No. 1, pp. 38-45, January-February 2017. http://ieeexplore.ieee.org/document/7819396/

[52] C. Anastasiades, T. Braun, V.A. Siris, “Information-Centric Networking in Mobile and Opportunistic Networks”, in: I. Ganchev, M. Curado, A. Kassler, “Wireless Networking for Moving Objects”, Lecture Notes in Computer Science, Vol. 8611, Springer, Cham, 2014.

[53] A. Carzaniga, M. Papalini, A. L. Wolf, “Content-based publish/subscribe networking and information-centric networking”, ACM SIGCOMM workshop on Information-centric networking (ICN '11), pp. 56-61, 2011.