Analyzing and Evaluating Information-Centric and Value-based Fog Service Architectures in Military Environments: The Phileas Simulator

Filippo Poltronieri[1,2], Cesare Stefanelli[1], Niranjan Suri[2,3], Mauro Tortonesi[1]
[1]University of Ferrara, Ferrara, Italy
{filippo.poltronieri, cesare.stefanelli, mauro.tortonesi}@unife.it
[2]Florida Institute for Human and Machine Cognition (IHMC), Pensacola, FL USA
{fpoltronieri, nsuri}@ihmc.us
[3]US Army Research Laboratory (ARL), Adelphi, MD USA
niranjan.suri.civ@mail.mil

Abstract

The accelerating evolution of Fog Computing technologies led to a growing interest towards their applicability to the Internet of Battlefield Things (IoBT). Efficient design for Fog Computing applications need to consider the optimal use of the available resources at the edge and the Cloud, switching from one or other depending on the current status, execution price, and user requirements. This is a complex task, as the optimization needs to consider the distributed and dynamic nature of the environment. There is the need to support investigation efforts by enabling researchers to experiment with Fog environments in a controlled and reproducible fashion. Unfortunately, most simulators do not implement a service model suited for Fog computing applications. This paper presents Phileas, a simulator that supports the definition of Fog services. Phileas allows to reenact the behavior of Fog services and evaluate different service policies and allocation strategies.

Index Terms

Internet of Things, Internet of Battlefield Things, Fog Computing, Value-of-Information.

I. Introduction

The accelerating evolution of Internet of Things (IoT) and Fog Computing related technologies in smart cities scenarios led to a growing interest towards their applicability to the Internet of Battlefield Things (IoBT). In fact, even if Fog Computing mostly finds its applicability in civilian scenarios, it can also be useful in different contexts. For example, it is possible to envision urban warfare or Humanitarian Assistance and Disaster and Relief (HADR) scenarios in which military personnels can exploit the capabilities of the existing civilian infrastructures to deploy application-specific IoBT services.

However the development of Fog Computing applications is a very difficult task, since it requires dealing with a large number of variables: devices and services placement, interactions with Cloud Computing platforms, and scalability issues in terms of both computational resources and number of users. Furthermore, most of these applications have high Quality of Service (QoS) and Quality of Experience (QoE) requirements, since they need to provide real-time information and low latency to the end users. At the other end, Fog Computing applications are necessary to tackle the deluge of data generated by IoT applications and devices, which is estimated by Cisco to reach 850 ZB by 2021 [1].

To address these issues, innovative solutions for Fog Computing applications started advocating the adoption of an "acceptable loss" perspective for Fog services, suggesting achievement of desired QoE levels by focusing on the processing and dissemination of the most valuable pieces of information for the end users [2]. With regards to this topic, *Value of Information (VoI)*, which measures the utility that a piece of information from the users' perspective, represents a compelling building block for the development of information prioritization solutions for processing and dissemination [3].

In addition, the efficient design of Fog Computing applications needs to consider the optimal use of the available resources at the edge and the Cloud [4], switching from one or other depending on the current status, execution price, and user requirements. This is a complex task, as the optimization needs to consider the distributed and dynamic nature of the environment. In fact, optimization tools need to be aware of the current status of resources and services to find optimal allocation for the tasks in order to maximize the the performance of the system.

There is the need to support investigation efforts by enabling researchers to experiment with Fog environments in a controlled and reproducible fashion. Unfortunately, most simulators do not implement a service model suited for Fog Computing applications, since this field is still relatively recent. Most of the existing solutions derive from Cloud-based simulators and do not address all the requirements of Fog Computing scenarios.

This paper presents Phileas, a simulator that supports the definition of Fog services. Phileas allows the reenactment of the behavior of Fog services and evaluates different service policies and allocation strategies

in order to find an optimal allocation for services and resources between the Cloud-IoT continuum. Phileas, unlike other simulators, makes use of the definition of VoI in the optimization process and aims at increasing Situational Awareness in civilian and military applications.

The rest of this paper is organized as follows. Section two discusses the Fog Computing and the Cloud-IoT continuum concepts. Section three describes the information maturity model adopted in the design of the simulator. Section four is about the design of the simulator. Section five provides an experimental evaluation of the results obtained by the simulations of a fictional Fog Computing scenario. Section six described other related works in the field of simulation for Fog Computing environments. Finally, Section seven concludes this paper and entails future works.

II. Fog Computing Scenario

In smart cities scenarios, a plethora of smart objects, sensors, vehicles, and personal devices provide capillary sensing functions at the edge of the networks. IoT applications leverage these data to perform many different functions such as providing emergency and/or healthcare services, effective traffic management, etc. [5] [6].

We envision a scenario in which IoBT applications can benefit of the existing IoT civilian infrastructure to install new IoBT specif devices and/or applications, for example, video surveillance and detection services for strategic assets. In fact, IoBT application should be capable of using the data produced by the smart sensing to elaborate context-specific services.

In most IoT applications analytics are implemented in the connected core of the network, where the collected raw data is stored and processed through sophisticated analytics in metropolitan area and Cloud level data centers. However, this approach brings significant problems from the point of view of network infrastructure utilization and application latency [7].

Fog Computing extends this scenario by allowing IT service developers and providers to allocate (a portion of the) information-processing tasks at the edge of the network, with the potential of significantly reducing the response latencies (and consequently improving the quality) of IT services and the reducing burden on the network infrastructure. As a result, Fog Computing represents a particularly attractive paradigm for the development of low latency, deeply immersive, and high-value-added IT services designed for digital citizens.

In this scenario, information processing tasks can be allocated either in the Cloud or on top of a plethora of different edge devices, including the aforementioned IoT gateways, Cloudlets or Micro-Clouds, and Multi-Access Edge Computing. In fact, researchers have started addressing this scenario using the *Cloud-IoT continuum* term [8].

Fig. 1 illustrates this scenario by highlighting the differences between the IoT/IoBT End-user, Fog, and Cloud layers of the network. More specifically, at the first layer, we have IoT End-users and IoT networks, usually represented as collections of sensors and other devices. These devices generate raw data, which are collected and analyzed at the upper Fog layer, where Micro-Cloud and IoT Gateway are responsible for a first processing of the gathered data. Finally, the last layer is represented by Cloud Computing platforms with storage and high computing capabilities. The information processed at these two upper layers is then returned to the End-users.

However, Fog applications need to be able to make the most out of the constrained bandwidth and computational resources at the edge of the network. This suggests the investigation and adoption of smart information-centric service models as opposed to a traditional Service Oriented Architecture, that are capable of selecting the most valuable from the end-user point of view.

III. The Adaptive, Information-centric, and Value-based Model for Fog Services

Researchers have recognized the concept of Fog Computing by proposing information-centric programming and service models for Fog applications, such as the one proposed in [9] which describes a Distributed Dataflow inspired by digital signal processing techniques and allows Business Process Execution Language

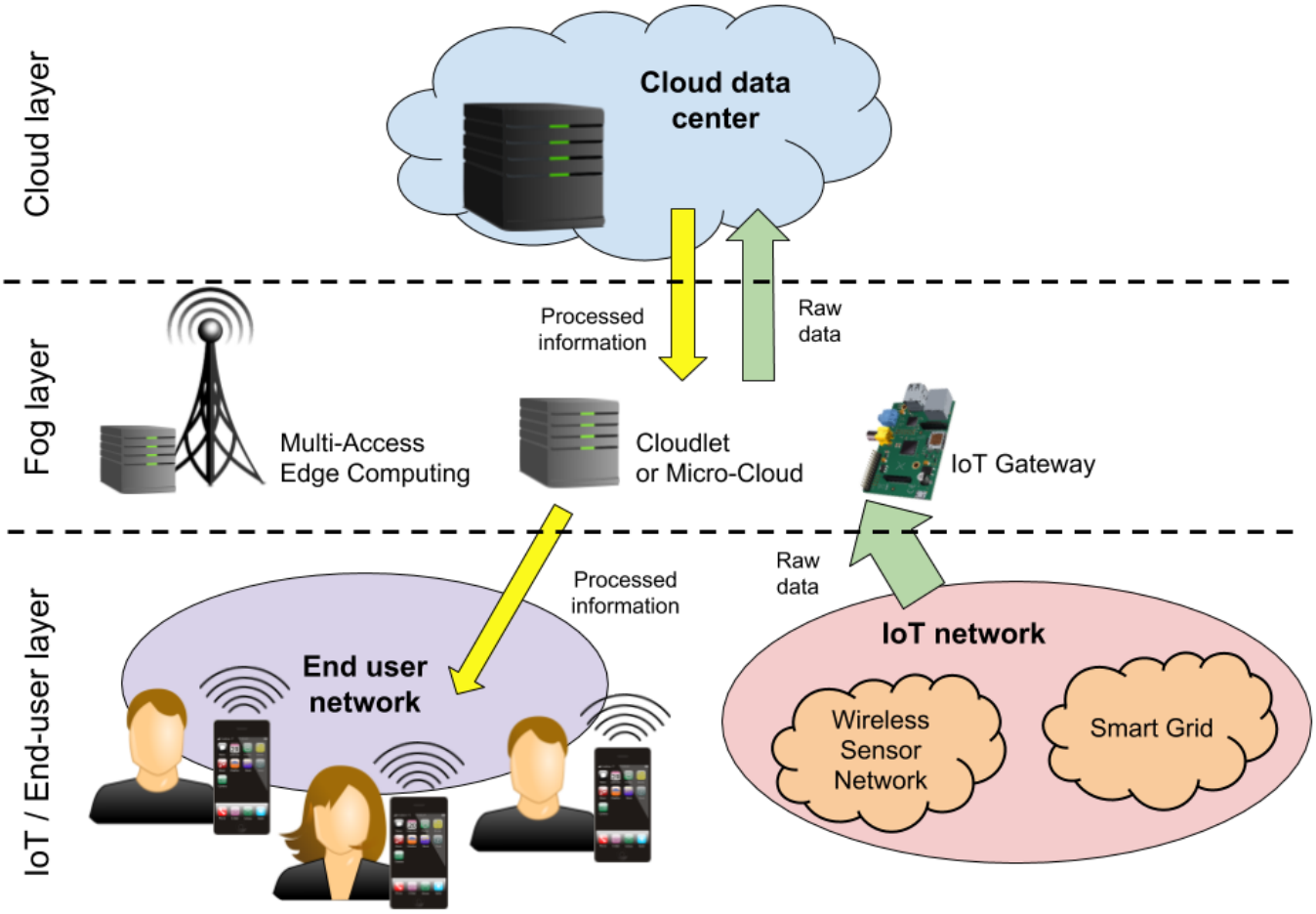

Fig. 1. Fog Computing Scenario.

(BPEL)-like service definition via scripting solutions. On the other hand, these solutions mostly provide techniques for service definition as orchestration of functions offered by different components, without explicitly addressing the mismatch between the formidable computational and bandwidth requirements of information processing tasks and the strict resource constraints that characterize Fog Computing environments.

Instead, the dynamic and resource-scarce nature of Fog environments suggests the adoption of a radically different perspective based on the "acceptable lossiness" concept and on innovative service models. The key idea is to realize services that are capable of automatically scaling their resource requirements to their current execution context while preserving high QoE levels.

More specifically, Fog services need to explicitly consider the different characteristics of Information Objects (IOs) and prioritize the processing and dissemination of the most important IOs, while discarding unimportant ones. In this context, the *Value-of-Information (VoI)* metric, which measure the utility that an IO provides to its recipient, represents a very interesting criterion for the purpose of information prioritization [3].

Along these guidelines, we decided to develop the *Adaptive, Information-centric, and Value-based (AIV)* model for Fog services. The model was developed within our experience with the Sieve, Process, and Forward (SPF) research project [2].

AIV proposes an *information maturity model*, which divides data and information processing stage in three different phases. The first processing phase regards raw data: input feeds of (typically sensing) data gathered from WSNs, smartphones, wearable devices, and other IoT devices in general. According to the

AIV model, raw data are analyzed by generic lower level processing functions to produce higher-order data constructs called Information Objects (IOs). In particular, IOs are generated by multiple aggregated and/or distilled raw data fused together to obtain more valuable information. Finally, the last processing phase makes use of the IOs to generate Consumption Ready Information Objects (CRIOs), which represent the information in its final stage, ready to be consumed by the users who requested it for. It is important to specify that in this information model, the generation of a raw data message does not automatically corresponds to the generation of an IO and the generation of an IO does not automatically produce a CRIO. In fact, in most of the cases the generation of an IO could require many raw data objects, and a CRIO can be the result of the aggregation of multiple IOs provided by many different services.

AIV is built on the top of this information maturity model and proposes an adaptive and content-base composition of different processing steps for Fog Services definition. In fact, according to AIV, Fog services implement processing functionalities to analyze the raw data generated by sensors and other devices at the edge to produce and distribute CRIOs to the end users.

In particular, the processing function of a Fog service is the result of the efforts provided by the coordinated orchestration of two different processing layers: *pipelines* and *services*. Pipelines gather and analyze raw data to produce IOs, often using low-level and/or service-agnostic algorithms and in some cases leveraging hardware, e.g., computational addons such as the Intel Movidius, or software, e.g., image processing libraries, resources that need to be preinstalled in the corresponding host device. Services instead, implement application-specific processing functionalities to further analyze the pipeline generated IOs to produce CRIOs, and usually can be migrated to other devices much more easily than pipelines.

Pipelines and services represent the basic building block of Fog services. They are meant to be composed in a loose, dynamic, and information-centric fashion. More specifically, Fog services can be defined as a sequence of pipelines and services that can be matched simply according to the content type of the messages generated. This loose definition of Fog services allows to rather easily support dynamic architectures in which the single instances of service components can be migrated to different devices along the Cloud-IoT continuum according to the current execution context (service requirements, resource availability, user preferences, etc.).

We chose to use this information maturity model, since it maximizes the opportunities to reuse processing components and generated results. In fact, pipelines implements basic processing functionalities that can be easily re-used by different services and pre-installed by Fog providers both in edge devices or in the Cloud. Instead, services that implement application-specific tasks need to be designed and implemented by Fog application developers.

IV. The Phileas Simulator

We explicitly developed the Phileas simulator to allow the reproducible evaluation of Fog applications in a realistic smart city environment. Phileas enables users to define Fog applications built on services that follow the AIV model illustrated in the previous Section.

A. Main Concepts

Phileas is based on 6 main concepts: *locations*, *data sources*, *devices*, *service types*, *service activations*, *and user groups*. Phileas models locations in a realistic fashion, associating geographical latitude and longitude coordinates to them. All entities modeled by Phileas, such as data sources, devices, and users are placed in a specific location. Geodesic distance between locations is calculated according to Vincenty's formula, which leverages an accurate ellipsoidal model of the Earth and is significantly more accurate than the simpler and more popular Haversine formula.

Data sources represent the equivalent of connected IoT sensors and continuously generate raw data of a given content type. For each data source, the message generation process can be defined by assigning 3 random variables that respectively control the time between subsequent message generations, i.e., the inter-generation time distribution, and the size and Value-of-Information attributes for each message generated.

Devices are the entities that host service components. Phileas models two types of devices: edge devices and Cloud platforms. Each edge device has a limited (and configurable) set of available resources, which are assigned to the service components running in the device in a weighted fair sharing fashion, according to the service components' resource requirements. Instead, Cloud platforms do not have any resource limitation and can always provide service components with the entire amount of resources they require.

According to the AIV model, Phileas allows building Fog services through a loose and information-centric composition of their building blocks. More specifically, each service component defines the content type and maturity level, e.g., raw data, IO, or CRIOs, of its input and output messages. Messages are then dispatched between information processing and consuming entities, i.e., data sources, service components, and end users, by matching their content type and maturity level attributes to the entities' interests.

In turn, service components are defined through the related type. By focusing on the definition of a common behavior shared by all service components of the same type, the service component type concepts facilitates the definition of multiple instances of the same service component. More specifically, service components define an amount of required resources. Following the AIV model, service components are automatically capable of scaling down their operations in case they cannot be assign the full amount of resources they require. In this case, incoming messages are rejected according to a linear message drop policy. For simplicity, at the moment of this writing Phileas models resources using a unidimensional scale. We might explore multidimensional modeling for resource sets, i.e., for instance considering CPU, storage, and communication resources as different dimensions, in future versions of the simulator.

In addition, each service component defines its processing policy, that is, the process of generating new messages from the analysis of received ones.

Service activations are events that define the time and location, i.e., the hosting device, for service component instantiations. Service activations are designed to be controlled by a separated component, that is currently under development. (In fact, the capability to evaluate the effectiveness of the decision making performed by a fully automated activation component is one of the main reasons for which we decided to develop Phileas in the first place.)

Finally, Phileas allows the definition of different user groups, a concept which is meant to model the presence of heterogeneous groups of Fog service users at a given (fixed) location. The number of users present in a group at any given time is stochastically modeled through a random variable. For each user group, Phileas allows defining the share of users within the group interested to a specific content type.

B. VoI Tracking and Processing

Phileas accurately models the VoI of each message, from its generation to its consumption. More specifically, Phileas has two different types of information producer entities can generate new messages: data sources and service components, respectively in response to the arrival of new raw data or to the processing of one or more lower-maturity messages.

Phileas tracks the VoI of each message exchanged from its origination to its consumption. Upon creation, each message is assigned an initial VoI value and a VoI decay profile that models the loss of VoI as time passes and the information travels from its originating source, according to the configuration of the entity that generated the VoI (data source or service component).

More specifically, Phileas allows assignment of two different types of VoI decay profiles for data sources and service component: a time decay and a space decay profile. In turn, each of those profiles supports 3 types of VoI obsolescence profiles: no decay, linear decay, and exponential decay.

At the moment of its generation, each new raw data message is assigned an initial VoI attribute, which is sampled from the VoI distribution modeling random variable associated to its data source.

Instead, the VoI of messages generated by service components (that can either be IOs or CRIOs, according to the service component definition) is calculated as function of the VoI of messages that need to be processed to generate the IO or CRIO message.

C. Communication model

Phileas adopts a pragmatic approach to communication modeling, oriented to allow the accurate evaluation of the Value-of-Information produced by the Fog services running in the simulated scenarios. More specifically, Phileas focuses on the adoption of relatively simple solutions that provide a reasonable accuracy in accounting message losses and communication delays while dismissing lower-level and protocol specific aspects such as transmission rate, interference modeling, etc.

Phileas assumes that latency in edge-to-Cloud communications to stochastically modeled according to a Gaussian mixture distribution. This is a realistic latency model, as proved by our recent research [10] that improves existing similar works in the literature, such as [11]–[13], by considering both Round-Trip Time (RTT) and Time-To-Live (TTL) parameters to obtain more realistic values. In addition, for simplicity, service components hosted in a Cloud platform are assumed to be always reachable from any device or user group.

Communications at the edge, i.e., involving data sources, service components hosted on edge devices, and/or end users, are modeled using a simplified propagation model. More specifically, Phileas adopts a propagation loss model based on the pre-calculation of the maximum communication distance. If the distance between communicating parties is smaller than the maximum communicating distance the message is considered successfully delivered, if not it is dropped. Let us point out that, while apparently simplistic, this mechanism is consistent with the propagation loss model used in many network simulators, such as Network Simulator 3 (NS3) [14], whose physical layer (PHY) implementation uses propagation loss models to evaluate if the signal power measured at the receiving device (also considering the antenna gain) is higher than the receiver sensitivity threshold. In that case, the simulator considers the packet as correctly received, if not, the packet is dropped.

We model the latency delay by sampling from a random variable with a long tail distribution. According to the latency analyses published in several recent research studies, this seems to be a simple but relatively realistic solution [15] [16] [17].

D. Implementation Insights

We implemented Phileas in the Ruby programming language, because of its excellent support for the definition of domain-specific languages, for its rapid application development, and for its access to a range of high quality libraries for geographic coordinate manipulation and random variable generation and sampling, such as the *geo_coord* and *erv* gems. Phileas is open source (MIT license) and is distributed on GitHub at the https://github.com/DSlG-UniFE/phileas URL.

V. Experimental Evaluation

In order to evaluate the Phileas effectiveness and potential in analyzing the performance of Fog applications based on the AIV model, we devised a realistic scenario. More specifically, we considered a fictional Fog Computing application running in the downtown area of Washington District of Columbia (DC), USA. We chose this location to setup different data sources, which will feed pollution, detection, face recognition, and traffic monitoring services running on several edge device and on the Cloud. Finally, to complete the scenario, we defined multiple user groups representing either civilians or military personnel) interested in receiving information from one or multiple services.

The pollution service modeled in this fictional scenario collects data from the in-range available smart-metering stations for the users interested interested in receiving notification about the pollution status nearby. This smart-metering service becomes essential for scheduling potential emergency evacuations and/or operations in both civilian or urban warfare scenarios. The detection service analyzes data samples gathered by detection sensors located in different areas of interest to monitor strategic assets and eventually triggers face recognition services to identify potential threats. In particular, the face recognition service uses Optical Character Recognition (OCR) capabilities, to analyze data collected from video's raw data, compare them with a database of known people, and inform the corresponding authorities. Finally, the

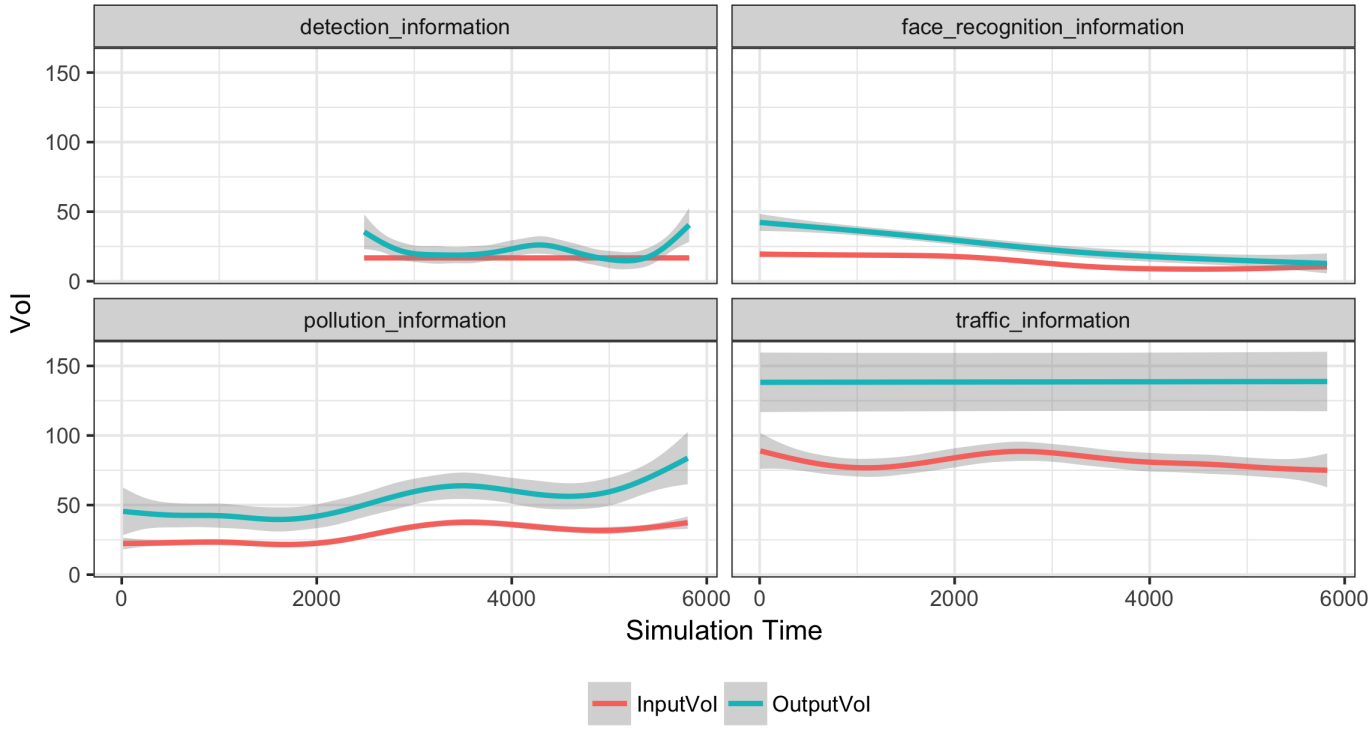

Fig. 2. Simulated VoI (Input/Output) trend for each different service.

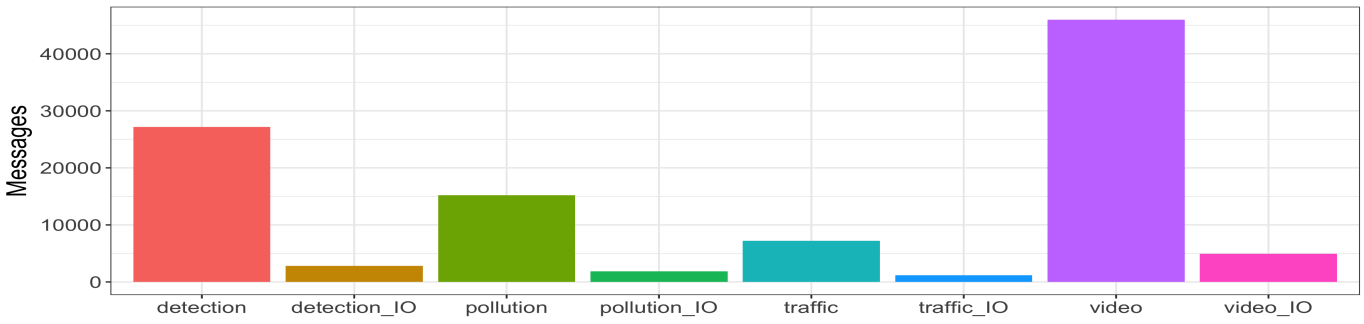

Fig. 3. Number of dropped messages.

traffic service gathers data from the cameras located at street intersections to determine the status of the viability and inform the users interested in it.

For the purpose of this experiment, we implemented the information processing logic of the services through a relatively simple fusion algorithm. More specifically, all the service components considered in the experiments define an incoming buffer of size b_t. When a message is received, it is momentarily put in the buffer. When the buffer is full, the service component generates a new message whose VoI is the average of the VoIs of the messages in the buffer measured at their receival time, multiplied by a constant service specific factor, called VoI multiplier. The service component then discards the buffer and its content, and creates a new one. Note that the size of the buffer is not constant but is sampled from a geometric distribution with $p = 0.9$. We believe that this scheme can approximate the behavior of a wide range of information-centric service components, that emit messages whose content is at least in part generated from the fusion of a sequence of messages.

We ran the fictional scenario described above in Phileas for the simulated duration of one day. Fig. 2 illustrates the amount of input and output VoI for each different service: detection, pollution, traffic,

and face recognition information. In detail, the input VoI represents the value of the raw data, while the output VoI represents the values of the CRIOs produced by each different services. Some services have higher VoI than others, in particular the traffic service, which per definition offers real-time/time-sensitive information. Furthermore, Fig. 2 also illustrates how the total output VoI is generally higher than the input VoI, thus remaining high even when the services allocated saturate the entire amount of resources available. More specifically, the activation of the detection service ten hours after the begin of the simulation does not affect significantly the VoI information of the other services.

Let us further analyze the results to illustrate how the AIV service model naturally enables the preservation of high VoI amounts by discarding lower value data when the amount of available computational resources is not sufficient to process all incoming raw data and IO messages. Fig 3 shows how the percentage of dropped messages changes in function of the service. As we expected and according to data discussed above, the higher number of dropped messages is registered for data regarding services with lower VoI. Furthermore, we can notice how, within the same service, the number of dropped IO messages is drastically lower compared to number of raw data messages, and thus according to the AIV definition. Moreover, comparing Fig. 3 with Fig. 2 one can notice that the VoI information is not affected by the dropping of the messages, since its value remains almost constant.

Finally, Fig. 4 illustrates the total amount of VoI generated in the fictional scenario with a curve color-coded according to the number of running services. To simulate the saturation of the available resources, we scheduled the activation of new services in order to evaluate output VoI variations. Once again the results show how the saturation of the available resources does not impact significantly the total amount of VoI provided to users. In fact, when new services saturate the available resources, the higher-VoI messages are prioritized over lower-VoI ones, and thus leading to the VoI conserving trend shown in Fig. 4.

VI. Related Works

Smart task allocation for Fog Computing environment had open research challenges. In fact, Fog Computing applications become more pervasive, thus calling for innovative solutions capable to monitor, allocate, and manage the available resources at the edge and the interactions between the Fog and Cloud Computing platforms. Simulation is a promising practice in this field of research, since it permits representation of scenarios involving a large number of variables. A relevant work is iFogSim [18] that aims to offer a complete toolkit for modeling allocation of task and resources in edge and Fog Computing environments. Another effort in the same direction is the one the described in [19], where the authors present a simulator based on Discrete Event System Specification to evaluate the impact of deploying Fog Computing applications. The evaluation process mainly involves computational power, processing delay, and scalability of the system in function of the number of the users and the services' allocation between the Fog and the Cloud. The authors concluded that using Fog in combination with the Cloud results in improved user experience and better performances. Another effort in the simulation direction is [20], where CrowdSensSim, a simulator to address Mobile Crowd Sense, is presented and applied to a smart

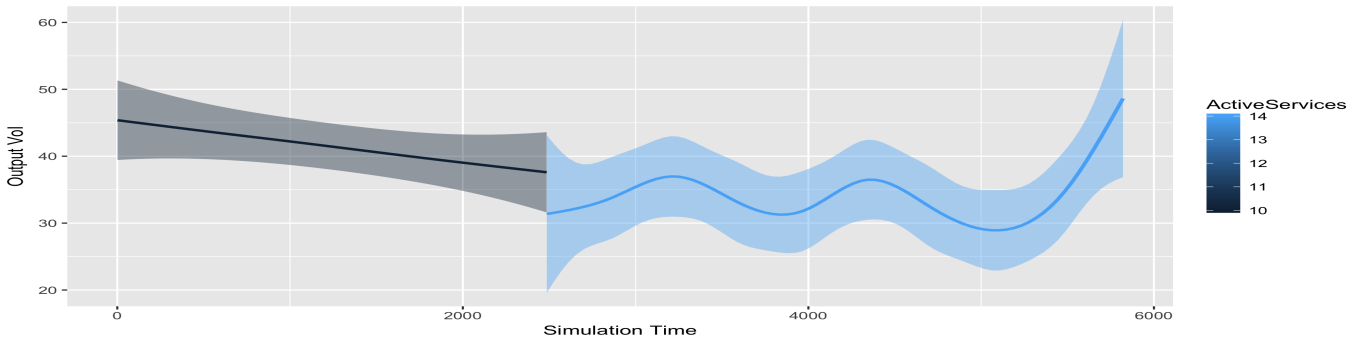

Fig. 4. Simulated VoI trend in function of the number of running services.

lighting application that aims at optimizing and reducing the costs due to public lighting. Even if this work does not explicitly address smart resource/task allocation between the Fog or Cloud environment, it contains valid efforts and considerations for designing smart city applications.

On the other hand, the majority of these works do not address task and resource allocation problems from the user perspective. Phileas, instead, aims at evaluating innovative service models and smart allocation solutions that optimize the total Value-of-Information produced by Fog services running along the IoT-Cloud continuum.

VII. Conclusions and Future Works

Phileas allows to evaluate the performance of Fog Computing applications from a user-centric utility perspective. Despite the Phileas project is still at the initial stage, preliminary results demonstrate the effectiveness of the AIV model in naturally maintaining a high VoI even when all available resources are saturated.

An important future work objective is the investigation of the VoI-optimal placement of information processing tasks along the Cloud-IoT continuum for Fog computing applications.

References

[1] "Cisco global cloud index: Forecast and methodology, 2016-2021." Cisco, 2018.

[2] M. Tortonesi, M. Govoni, A. Morelli, G. Riberto, C. Stefanelli, N. Suri, "Taming the IoT Data Deluge: An Innovative Information-Centric Service Model for Fog Computing Applications," *Future Generation Computer System*, in press.

[3] N. Suri, G. Benincasa, R. Lenzi, M. Tortonesi, C. Stefanelli, L. Sadler, "Exploring value of information-based approaches to support effective communications in tactical networks," *IEEE Communications Magazine*, vol. 53, no. 10 (Special Feature on Military Communication, pp. 39–45, 2015.

[4] M. Mukherjee, l. Shu, D. Wang, "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges," *IEEE Communications Surveys & Tutorials (Early access)*, March 2018.

[5] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641 – 658, 2018.

[6] J. Liu, J. Li, L. Zhang, F. Dai, Y. Zhang, X. Meng, and J. Shen, "Secure intelligent traffic light control using fog computing," *Future Generation Computer Systems*, vol. 78, pp. 817 – 824, 2018.

[7] A. Papageorgiou, B. Cheng, and E. Kovacs, "Real-time data reduction at the network edge of Internet-of-Things systems," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 284–291.

[8] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, 2018.

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

[10] W. Cerroni, L. Foschini, G. Y. Grabarnik, L. Shwartz, and M. Tortonesi, "Estimating Delay Times Between Cloud Datacenters: A Pragmatic Modeling Approach," *IEEE Communications Letters*, vol. 22, no. 3, pp. 526–529, March 2018.

[11] S. Secci, P. Raad, and P. Gallard, "Linking virtual machine mobility to user mobility," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 927–940, Dec 2016.

[12] L. M. Vaquero, S. S. Lor, D. Audsin, P. Murray, and N. Wainwright, "Sampling isp backbone topologies," *IEEE Communications Letters*, vol. 16, no. 2, pp. 272–274, February 2012.

[13] T. Mizrahi and Y. Moses, "On the behavior of network delay in the cloud," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2016, pp. 875–876.

[14] K. Wehrle, M. Gne, J. Gross, *Modeling and Tools for Network Simulation*. Springer, Berlin, Heidelberg, 2010.

[15] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, "WiFi can be the weakest link of round trip network latency in the wild," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[16] K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, and T. Moscibroda, "Characterizing and Improving WiFi Latency in Large-Scale Operational Networks," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobySys '16)*, 2016, pp. 347–360.

[17] T. Høiland-Jørgensen, P. Hurtig, and A. Brunstrom, "The Good, the Bad and the WiFi: Modern AQMs in a residential setting," *Computer Networks*, vol. 89, pp. 90 – 106, 2015.

[18] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Journal of Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, September, 2017.

[19] M, Etemad, M. Aazam, M. St-Hilaire, "Using DEVS for modeling and simulating a Fog Computing environment," in *2017 International Conference on Computing, Networking and Communications (ICNC), Santa Clara, CA, USA*, Jan 2017.

[20] C. Fiandrino, A. Capponi, D. Kliazovich, P. Bouvry, F. Granelli, S. Giordano, "CrowdSenSim: a Simulation Platform for Mobile Crowdsensing in Realistic Urban Environments," *IEEE Access*, pp. 3490–3503, 20 February 2017.

APPENDIX

Example configuration for a data source:

```
data_sources(
  1 => {
    voi_dist: { distribution: :exponential, args: { rate: 0.1 } },
    message_size_dist: { distribution: :exponential, args: { rate: 0.1
     ↪ } },
    time_between_message_generation_dist: { distribution: :exponential,
     ↪ args: { rate: 0.1 } },
    output_content_type: :video,
    location_id: 1,
    time_decay:  { type: :linear, halflife: 1000.0 },
    space_decay: { type: :linear, halflife: 1000.0 },
  },
)
```

Example configuration for a service type:

```
service_type(
  1 => {
    input_content_type: :pollution,
    input_message_type: :raw_data,
    output_content_type: :pollution_IO,
    output_message_type: :io,
    resource_requirements: 35.0,
    time_decay:  { type: :linear, halflife: 1000.0 },
    space_decay: { type: :linear, halflife: 1000.0 },
    processing_policy: {
      type: :aggregation,
      aggregation_window_size_dist: { distribution: :discrete_uniform,
       ↪ args: { min_value: 1, max_value: 10 } },
      aggregated_message_size_dist: { distribution: :discrete_uniform,
       ↪ args: { min_value: 1024, max_value: 2048 } },
      voi_multiplier: 4.0,
    },
  },
)
```

Example of service activation:

```
service_activations(
  1 => { type_id: 1, at: { time: start_time, device_id: 1 } },
)
```

Example configuration for a user group:

```
user_groups(
  1 => {
    user_dist: { distribution: :exponential, args: { rate: 0.3 } },
    location_id: 3,
    interests: [
      { content_type: :pollution_information, share: 0.8 },
      { content_type: :detection_information,  share: 0.2 },
      { content_type: :face_recognition_information, share: 0.7 },
    ]
```

```
    },
)
```