Università
degli Studi
di Ferrara

DOTTORATO DI RICERCA IN
SCIENZE DELL'INGEGNERIA
CICLO XXXIII
COORDINATORE Prof. Stefano Trillo

Value-of-Information Middlewares for Fog and Edge Computing

Settore Scientifico Disciplinare ING-INF/05

Dottorando
Ing. Filippo Poltronieri

Tutore
Chiar.mo. Prof. Ing. Cesare Stefanelli

Cotutore
Prof. Ing. Mauro Tortonesi

Anni 2017-2020

Sommario

Con i termini Fog ed Edge Computing si indicano dei paradigmi computazionali che, spostando l'elaborazione dei dati IoT nelle prossimità sia dei dispositivi che degli utenti, mirano a fornire servizi a bassa latenza, immersivi e real-time. Fog ed Edge Computing trovano applicazione in contesti Smart Cities, dove è possibile sfruttare la capacità computazionale di gateway IoT, Cloudlet e Base Station per elaborare parte dei dati generati dall'IoT direttamente ai margini della rete. L'adozione dei paradigmi di Fog ed Edge Computing è tuttavia complessa in quanto pone una serie di sfide tra cui il processamento dell'enorme mole di dati generati dall'IoT, la presenza di un numero limitato di dispositivi altamente eterogenei e con capacità computazionali scarse, requisiti di servizio altamente dinamici e reti con caratteristiche eterogenee. Per garantire i requisiti stringenti di bassa latenza, soluzioni per Fog ed Edge Computing devono essere in grado di utilizzare al meglio le scarse risorse a disposizione, gestendole al meglio. Se questi paradigmi sono oggetto di ampie ricerche, vi è la necessità di investigare soluzioni innovative che consentano di gestire l'enorme mole dati IoT e permettere una concreta applicazione di Fog ed Edge Computing. Questa tesi propone middleware innovativi in grado di fornire soluzioni complete per fronteggiare al meglio le caratteristiche altamente dinamiche di scenari Smart Cities, fornendo metodologie e strumenti per allocare e distribuire servizi tra le risorse a disposizione, monitorare lo stato delle risorse e modificare prontamente la loro configurazione. Come criterio innovativo per la prioritizzazione dei dati IoT per processamento e disseminazione, questa tesi adotta il concetto di Value-of-Information (VoI), nato come estensione della Teoria dell'Informazione di Shannon e applicato in ambiti decisionali. A tal fine, questa tesi propone politiche di gestione delle informazioni che consentono di realizzare servizi modulari e facilmente (ri-)componibili e tecniche di ottimizzazione innovative che ben si adattano a questi servizi. Inoltre, i middleware presentati in questa tesi integrano il concetto di VoI sia a livello di servizio che a livello di gestione per selezionare le informazioni più preziose per l'elaborazione e la diffusione, riducendo così il carico computazionale e garantendo una gestione ottimale dei dispositivi e della rete. Le ricerche presentate in questa tesi sono il risultato della collaborazione con istituti di ricerca internazionali e di un periodo di ricerca trascorso presso il Florida Institute for Human and Machine Cognition (IHMC), FL, USA.

Summary

Fog and Edge Computing aim to deliver low-latency, immersive, and powerful services by processing information close to both devices and users. This is well suited for IoT applications in Smart City, where IoT gateways, Cloudlets, Base Stations, and other computational nodes can process (part of) the data generated by the multitude of IoT sensors directly at the edge of the network. However, the implementation of Fog and Edge Computing is challenging because it requires to deal with a (limited number of) constrained devices, dynamic services' requirements, and heterogeneous network conditions. Differently from the Cloud, where computational resources are supposed to be unlimited, Fog and Edge services should be capable to adapt to scarce and constrained resources and deal with the deluge of IoT data. To facilitate the adoption of Fog and Edge Computing this thesis proposes innovative middlewares capable of providing comprehensive solutions to address the highly dynamic characteristics of these environments. These middlewares provide functions to allocate and distribute Fog and Edge services among the available computational devices, monitor the status of the environment, and promptly modify their configuration. To deal with the IoT data deluge this thesis investigates the interesting criterion of Value-of-Information (VoI). Originally born as an extension of Shannon's Information Theory for decision making science, researchers have studied VoI as an information management tool to select and prioritize information processing and dissemination. For this purpose, this thesis proposes the adoption of information management policies allowing the definition of service components, composable software modules that can be chained to create larger and more complex services. In addition, the middlewares presented in this thesis leverage the promising concept of VoI to select only the most valuable piece of information for processing and dissemination and to scale computational workload in an automated and lossiness fashion. This would enable to reduce the computational and network load and to propose innovative methodologies to optimize the available resources. The research efforts presented in this thesis are the results of the collaboration with international institutes and a research period at the Florida Institute for Human and Machine Cognition (IHMC), FL, USA.

Contents

Chapter 1

Introduction

The widespread adoption and the continuous evolution of Internet-of-Things (IoT) related technologies open to a multitude of new application scenarios that would leverage on services capable of processing IoT data for realizing immersive and real-time applications [1, 2]. This would provide enormous benefits to the end-users of these applications, which will be capable of exploiting the data generated by a multitude of IoT sensors and smart devices to obtain important information for several use-cases such as Smart Cities, smart transportation, and smart agriculture [3, 4].

A common approach for supporting the requirements IoT applications is to adopt Cloud-centric solutions, where the IoT data generated at the edge is sent to Cloud data-centers for processing. However, considering the notable amount of IoT data, a Cloud-centric approach would likely saturate the available network bandwidth and will result in services with poor performance and high latency. In fact, if Cloud-based solutions can offer huge and scalable computational resources they are not capable to satisfy the low latency and real-time requirements of IoT applications.

To address the limitations of Cloud-centric models, several computing paradigms have been proposed during the last decade such as Fog Computing [5] and Edge Computing [6, 7]. Even if they present some differences, all these paradigms leverage the common idea to bring computation close to both users and devices to *exploit their proximity and reduce latency*. More specifically, Edge Computing is a concept that advocates to enable the processing of data at the edge of the network, in close proximity of where the IoT data is generated by deploying edge devices for performing computational tasks. On the other hand, Fog Computing extends the Cloud Computing model to the edge for providing a set of functionality to distribute the data processing between three different layers: a Cloud layer, a Fog layer, and an IoT/Edge layer. Differently from Edge Computing, Fog Computing clearly depicts the connection and the cooperation of these layers, while Edge Computing only refers to the edge layer. Therefore, if conceptually Fog and Edge Computing are

different, they are widely interchanged in the literature to indicate the possibility of doing part of processing directly at the edge.

Fog and Edge Computing paradigms would enable to exploit IoT and edge devices, for elaborating IoT data close to where is generated and where information will be consumed, thus saving Internet bandwidth, reducing information latency, and providing high levels of Quality-of-Service (QoS) and Quality-of-Experience (QoE) [8, 1]. To realize this, academia and industry widely investigated Fog and Edge Computing during the last years [9, 10, 11]. Some standards have been proposed such as the OpenFog reference model for Fog Computing [12] and multi-access Edge Computing (MEC) for telecommunication networks [13, 14]. However, they mainly address architectural models and specific use-cases without dealing with the importance of programming models and the IoT data deluge. Therefore, Fog and Edge Computing still remain rapidly evolving industrial and academic research topics [15].

At the time of this writing, the adoption of Fog and Edge Computing solutions presents several challenges that still need to be addressed. In fact, Fog and Edge Computing have to deal with scarce computational capabilities, limited bandwidth, services migration, and users and devices mobility. In addition, computing devices at the edge often present heterogeneous characteristics both from a hardware and software perspective, thus making the management of such resources more challenging. Also, network links at the edge can present different characteristics in terms of availability, latency, and bandwidth that need to be considered in the management of applications. These peculiarities make Fog and Edge Computing very dynamic environments, thus making their practical adoption even more challenging.

To effectively exploit the resources available at the edge, there is the need for architectural solutions capable of coordinating and distributing the processing of the deluge of IoT data on the available computing resources. Furthermore, to tackle the high dinamicity of these environments, Fog and Edge Computing solutions must provide tools for enabling frequent re-configurations of network and services to guarantee high levels of QoE and QoS and to deal with the aforementioned challenges. Finally, Fog and Edge Computing solutions have to address the IoT data deluge by proposing innovative methodologies capable of filtering the data to be processed and disseminated, thus avoiding the saturation of available computing power and network bandwidth at the edge of the network. Therefore, the efficient adoption of Fog and Edge Computing calls for novel methodologies and tools to address the challenges of these environments and enable matching the low latency requirements of IoT applications.

This thesis contributes to Fog and Edge Computing by proposing innovative methodologies and tools for the definition of middlewares that would allow service providers and users to exploit the promises of Fog and Edge Computing paradigms.

To achieve such a goal, these middlewares provide the essential functionality to coordinate and distribute the processing of services on a pool of fog and edge devices by selecting the most suitable allocations considering the current status of the environment and the proximity of IoT data sources and end-users.

As innovative resource management methodology this thesis investigates *Value of Information (VoI)* maximization, i.e., the maximization of the utility that Information Objects (IOs) deliver to end recipients. VoI has originated from the seminal research in [16] as an extension to Shannon's Information Theory and it has been a widely explored topic in decision sciences. Recently, VoI based solutions were proposed in several application areas [17, 18, 19, 20]. However, while interesting results have been achieved so far, they mostly followed *objective approaches* that calculate VoI of a message according to its content and the distance between time and location of information generation and receival. Very few works considered *subjective approaches* that take into account different utility models per each user when estimating VoI [21, 22]. To fill this gap, this thesis makes a step forward for effectively integrating VoI into Fog and Edge resource management.

From an architectural perspective, this thesis presents the adoption of a service model that defines the concept of "service components": loosely-coupled and composition-friendly software modules that can be quickly chained to create more comprehensive services, and just as quickly rearranged in case of need. Service components exploit the VoI concept in the processing stage by seamlessly prioritizing the processing of the most valuable data, thus reducing the amount of information to be processed and disseminated. This would allow to optimize available computational resources and deliver the expected QoS and QoE for most valuable services operating in Fog and Edge computing environments.

On top of that, this thesis presents the SPF platform as the reference implementation of the illustrated service model, and the Phileas simulator, a simulation tool that would enable to simulate middlewares in realistic Fog and Edge Computing scenarios. On the one hand, SPF is a fully functioning platform developed to manage and distribute the execution service components on processing device (gateway in the SPF terminology) that can be located both at the edge of the network or in Cloud data centers. On the other hand, Phileas is a discrete event simulator built upon the experience of SPF to enable the reproducible evaluation of middlewares in large scale and realistic scenarios. SPF and Phileas would be the buildings blocks to realize a family of middlewares capable of tackling the challenges of Fog and Edge Computing both from an application and a network perspective. With these middlewares, this thesis wants to propose solutions that can address the high dynamicity of Fog and Edge Computing by leveraging on tools enabling to quickly react to the environment's changes via a prompt reconfiguration.

With this goal, this thesis defines a formal notation for VoI allowing to model the allocation of service components on computational devices and to measure the VoI utility derived from a certain allocation. This enables to estimate the total VoI delivered in a specific time-window and to plan re-configuration of services and networks. Furthermore, this thesis considers the VoI services' modeling as the basis to realize an intelligent use of computational and network resources in Fog and Edge Computing. To achieve this goal, this thesis describes different optimization methodologies such as simulation-based optimization, heuristics, and the adoption of a continuous optimization framework leveraging Deep Reinforcement Learning (DRL), which continuously monitors the current status of the environment for reacting to devices or network failures by proposing a different configuration for service components.

Moreover, to tackle such challenging environments this thesis supports the idea that only comprehensive approaches considering both the application and network perspective can result in the efficient management of Fog and Edge Computing. In fact, by adopting comprehensive solutions it is possible to have a wider and holistic perspective on how to optimally manage the available resources. Finally, to address the problem of IoT data deluge, the presented middlewares integrate VoI methodologies and tools into the management of both networks and applications, thus allowing to filter and to prioritize the data and information to be disseminated and processed. For this purpose, these middlewares manage services and resources in a way that the VoI delivered to the end-users of applications is maximized. This would enable to achieve better management of the scarce and coarse resources available at the edge, thus enabling effective Fog and Edge Computing.

The research efforts presented in this thesis have been published and submitted in the proceedings of international conferences and journals. The research activity has been conducted with international research institutions during my Ph.D. and within a profitable research period and collaboration with the Florida Institute of Human and Machine Cognition (IHMC), FL, USA.

The remainder of this thesis is organized as follows. Chapter 2 provides background on Fog and Edge Computing for describing the environments in which this thesis contributes to. Then, Chapter 3 presents the concept of VoI for Fog and Edge Computing and it describes the Additive Information-centric and Value-based (AIV) information model for defining service components and their allocation on devices.

Chapter 4 presents the SPF platform and the Phileas simulator as methodologies and tools used for the implementation of more comprehensive middlewares. The former is the reference implementation of the AIV model, while the latter is a discrete event simulator to reenact the behavior of middlewares operating in realistic Fog and

Edge Computing scenarios.

Chapter 5 gives a characterization of VoI optimization models to maximize the end-users utility of Fog services. In particular, the practical adoption of optimization models is discussed for both offline and online resource management inquiring that such exact models require simulation-based optimization. As a viable alternative, a distance based heuristic for approximation is proposed and evaluated with promising results.

Then, investigating deeply into VoI optimization, Chapter 6 analyzes Reinforcement Learning (RL) as a novel technique to integrate VoI methodologies and tools into a continuous optimization framework. With this goal, this Chapter presents the FogReinForce algorithm for learning the configuration of Fog Computing services that maximizes the total VoI delivered in a specific time-window.

Leveraging on the described methodologies and tools, Chapter 7 presents HORNET, a proof-of-concept middleware for realizing the holistic management of Fog Computing. In particular, HORNET aims to integrate VoI methodologies and tools into the management of both the network and applications to seamlessly prioritize the most valuable information and services. To reach such a goal, HORNET proposes a multi-layer routing approach that exploits multiple routing options at different abstraction levels. The proposed approach is presented and then validated in a small-scale real testbed and in a simulated larger testbed within Phileas.

On the other hand, Chapter 8 introduces the concept of Humanitarian Assistant and Disaster Relief (HADR) as a possible use-case for Fog Computing. HADR operations usually occur after a natural disaster and involve the collaboration of multiple rescuers, e.g. firefighters and medical operators. Considering a Smart City as the set of a HADR scenario, this Chapter reports the challenges that HADR operators may face in interacting with the existing Smart City Infrastructure. To facilitate them to exploit IoT assets already presented in Smart Cities, this Chapter presents an extended architecture for the SPF platform that aims to achieve enhanced interoperability with IoT assets for both civilian and military operators.

Within the same important topic, Chapter 9 takes a wider and holistic perspective and presents the ACESO middleware for Smart Cities. ACESO is a proof-of-concept middleware that aims to provide full support for HADR by design. In particular, this Chapter extends the VoI formulation presented in Chapter 5 for modeling "location-aware services" that would help rescuers involved in the HADR operations. Furthermore, this Chapter discusses also the importance of secure information exchange between HADR teams (both civilian and military) involved in the operations. Finally, this Chapter provides a detailed description of a fictional HADR scenario that involves multiple rescue teams and evaluates the ACESO middleware within the Phileas Simulator.

Chapter 2

Background

This Chapter aims to provide a background on Fog and Edge Computing as compelling computing paradigms to deal with the IoT data deluge and provide low-latency and immersive services. With some differences from one to the other, Fog and Edge Computing advocate to bring computation close to both users and devices, thus providing reduced latency and enhanced performance. Fog Computing proposes an architecture composed of three main layers: a Cloud layer, a Fog layer, and an IoT/Edge layer. Therefore, Fog Computing aims to consider the whole picture when distributing the processing of data. On the contrary, Edge Computing only considers the IoT/Edge layer into account when dealing with the processing of IoT data.

Bringing the processing close to the edge network would allow services to exploit the computational resources located in proximity of IoT sensors and users, thus avoiding the high latency required for uploading data to Cloud data centers. Fog and Edge Computing are the enabling factors for delivering services with strict, low-latency and real-time requirements.

2.1 Fog and Edge Computing

IoT applications should provide a wide range of low-latency and real-time services for several scenarios such as smart city, intelligent transportation, smart agriculture, and so on. This is made possible by the deluge of IoT data collected from sensors, smart appliances, and other devices located almost anywhere. However, this enormous amount of data makes Cloud-centric solutions inefficient for running IoT applications with strict requirements. In fact, Cloud-centric solutions suffer from high latency curse due to fact that they require IoT data to be uploaded to Cloud data centers for processing. Other limitations that Cloud-centric solutions face are bandwidth consumption (the amount of IoT data could be greater than the available bandwidth), intermittent connectivity with the IoT/Edge layer, and

other aspects related to constrained nature of IoT devices which might not be able to transmit data to the Internet.

To overcome the limitations of Cloud-centric solutions, Fog and Edge Computing paradigms have been proposed as viable alternatives for processing data close to both devices and users. The main objective of Fog and Edge Computing paradigms is to allow service providers to exploit IoT and edge devices, for elaborating IoT data close devices or users or both, thus saving Internet bandwidth, reducing information latency, and providing enhanced levels of QoS and QoE. Furthermore, Fog and Edge Computing enable to reduce the dependency with the Cloud in delivering services to users.

More specifically, Fog Computing extends the Cloud Computing model to the edge of the network by envisioning a multi-layer architecture on which different sort of devices can contribute to the data processing. A general Fog Computing architecture depicts a Cloud layer, a Fog Layer, and an IoT/Edge Layer. The Cloud layer is to represent Cloud data-centers either public or private or hybrid where it is possible to allocate part of the computational processing of the IoT data. In Fog Computing, this layer is usually restricted to the processing of services requiring high computational power or for batch services, which analyze raw-data coming from the IoT but do not require compliance with low-latency and real-time requirements. Also, this layer is depicted at the top to indicate that Cloud resources can be located in undisclosed locations, far apart from the IoT data is generated. Just below the Cloud layer is located the Fog layer, populated by different types of computational devices deployed at base stations or other locations with a direct and broadband access to the Internet. Finally, at the bottom layer of a Fog architecture is the Edge / IoT layer where sensors, IoT devices, and users reside. It is worth specifying that in some representation the IoT and the Edge layers are two different entities and that the Edge layer is located above the IoT layer.

On the other hand, Edge Computing considers only the IoT / Edge layer located at the burden of the network. Therefore, Edge Computing does not take into account a fog or cloud layer when distributing the processing of IoT data or other computational tasks. As Fog Computing, Edge Computing aims to deliver low-latency services by operating close to device. It is worth noting that even if conceptually different, Fog and Edge Computing are widely interchanged in the literature because of the wide similarities and the common objectives. In addition, the term Edge Computing is predominant in telecommunication, where ETSI has adopted multi-access Edge Computing (MEC) as the de facto standard for allowing application developers and content providers to provide services at the edge. Instead, Fog Computing is under the umbrella of the OpenFog consortium, a partnership of universities and tech industries for the standardization and the promotion of Fog

Computing. ETSI and OpenFog consortium signed a collaboration for developing fog-enabled mobile edge applications and technologies [23].

Finally, it is worth noting that, in the context of this thesis, referring only to Fog Computing instead of Fog and Edge Computing is not a lack of generality. In fact, this thesis refers to Fog to indicate a more general perspective, considering Fog as a superset of Edge Computing, which would allow middlewares to consider the whole Cloud-IoT continuum for services and resources management. This is to present solutions that would work in both contexts, leaving to the particular scenario the choice of what set of resources best fit with the environment's needs.

2.2 Challenges

If Fog and Edge Computing are promising concepts, developing applications for Fog and Edge Computing is a very difficult task. It requires to deal with a large number of variables: devices and services placement, interactions with Cloud Computing platforms, and scalability issues in term of both computational resources and number of users. Furthermore, most of these applications have high QoS and QoE requirements, since they need to provide real-time information and low latency to the end-users.

Fog and Edge solutions encounter several challenges such as the deluge of IoT data to analyze, scarce computational capabilities at the edge, limited bandwidth, services migration, and users and devices mobility. Firstly, computing devices deployed along the Cloud-IoT continuum present heterogeneous characteristics both from a hardware and software perspective, thus making the management of such resources more challenging. Secondly, the management of fog and edge networks raises other challenges due to the constrained and heterogeneous nature of the environments in which they operate. IoT sensors and smart devices operate in multiple Wireless Sensor Networks (WSN) using a multitude of low-power wireless communications protocols such as Bluethooth LE, IEEE 802.15.4, Long Range (LoRA), etc. At the other end, users interact with the network infrastructure using other communication protocols such as WiFi, LTE/4G, 5G, etc. On top of that, Fog and Edge Computing need to consider the high mobility of users and terminals, e.g devices deployed connected vehicles, which causes frequent communication disruptions and wide variability in channel performance.

Resource discovery is another important challenge for Fog and Edge Computing solutions that need to be aware of the availability of both raw-data sources and computational devices where to allocate the processing on. Furthermore, aside for the availability of devices, resource discovery for Fog and Edge Computing should also describe the capacity in terms of computing power and storage to provide

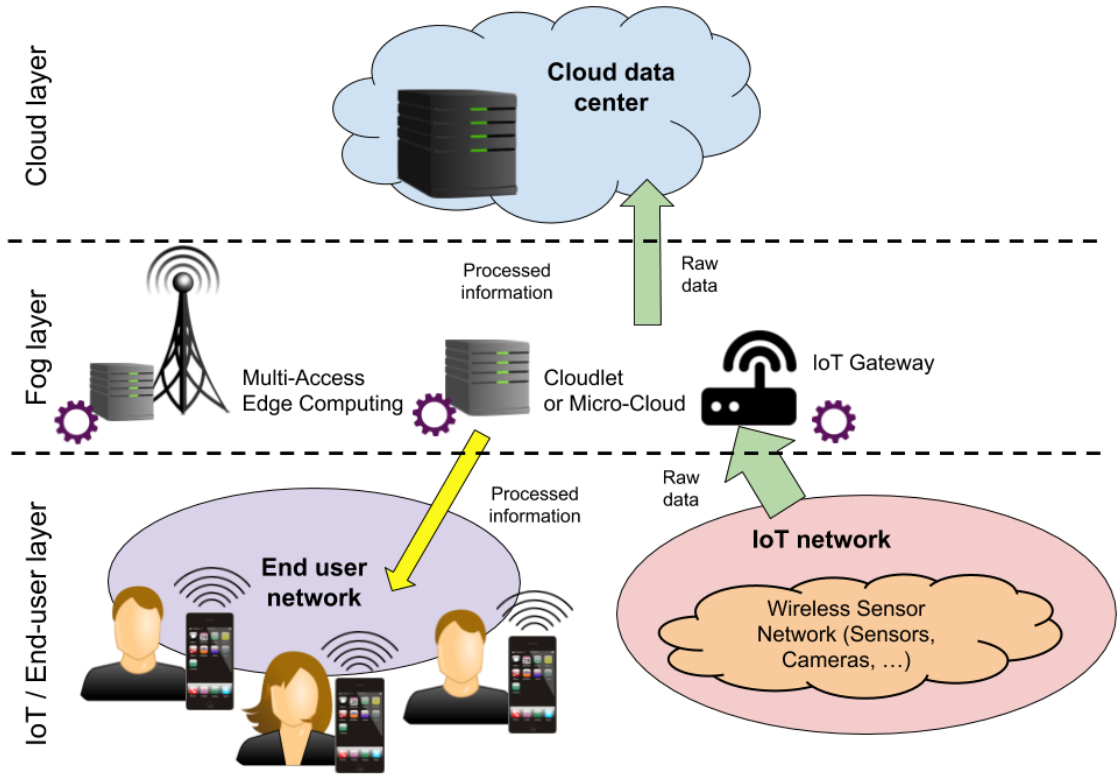


Figure 2.1: Fog Computing Scenario. Raw data collected are elaborated and processed

valuable insights for resource and service management tools.

The efficient design of Fog and Edge Computing applications needs to consider the optimal use of the available resources at the edge and the Cloud, switching from one other depending on the current status, execution price, and user requirements. This is a complex task for Fog and Edge management tools that need to consider the distributed and dynamic nature of the environment. In addition, management and optimization tools need to be aware of the current status of resources and services to find the optimal allocation for tasks that maximizes the performance of the system.

Despite these challenges, Fog and Edge Computing applications are necessary to tackle the deluge of data generated by IoT applications and devices. Therefore, there is the need to investigate solutions that would enable effective Fog and Edge Computing both from a service provider and an end-user perspective.

2.3 Smart Cities Scenarios

Smart Cities are one of the most interesting application scenario for Fog and Edge Computing. Their networking infrastructure is characterized by an edge, in which a plethora of smart objects, sensors, vehicles, and personal devices provide capillary sensing functions, and by a connected core, where the collected raw data is stored

and processed through sophisticated analytics in metropolitan areas and Cloud level data centers.

From the communication perspective, there are two main types of edge networks: IoT networks, that connect IoT devices one each other, and end-user networks, that connect users with the rest of the network using wide range wireless communication protocols such as WiFi, LTE/4G, and 5G.

Typically, IoT devices are deployed in groups to form sophisticated sensing systems and communicate based on IoT networks exploiting a multitude of short-range and low-power wireless communications protocols, e.g., IEEE 802.15.4, Bluetooth LE, and NFC. Then, IoT networks are connected to other wider networks (or even the Internet) through one or more IoT gateway devices. IoT gateways typically have significantly better computation, storage, and communication capabilities if compared with IoT devices.

End-user networks are typically heterogeneous networks formed by a set of overlapping medium-range wireless networks of different types, including cellular networks with macro and pico cells, Wi-Fi networks, etc. Device-to-device communications, such as WiFi-direct and LTE-direct, further expand communication opportunities by allowing personal devices to share information of common interest when they are in proximity.

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 reducing the 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. [24]. Also note that users' smart personal devices can, and typically do, perform a dual role: they access Fog Services and at the same time they operate as raw data sources thanks to the recent and remarkable developments in their sensing capabilities.

However, the implementation of Fog Computing solutions in Smart City environments represents a very difficult task for three main reasons. First and foremost, processing a massive amount of raw data generated by the IoT (predicted to increase to as high as 850 ZB per year by 2021 [25]) with the scarce computational and energy resources available to edge devices represents a daunting challenge. In addition, the significant mobility of users and terminals [26] causes frequent communication disruptions and wide variability in channel performance. Finally, edge resources are

very dynamic and heterogeneous. This calls for novel solutions that explore trade-offs between processing speed and accuracy to best exploit the resources available along the Cloud-things continuum.

2.4 State-of-the-Art

Fog and Edge Computing has been widely investigated as promising computing paradigms to bring computation close to the edge side of the network. While there are some controversy on when to use one term instead of the other, both Fog and Edge Computing related paradigms share the goal of providing reduced latency and better performance in general by exploiting (in different parts) the resources at the edge of the network. This Section aims to report the state-of-the-art in the definition of Fog and Edge Computing.

Fog Computing was firstly discussed in [5], in which Bonomi et al. described Fog Computing as a highly virtualized platform for providing computing, storage, and networking services between the Cloud and the edge of the network. A similar definition is provided in the white paper from Cisco [27], in which Fog Computing is defined as a paradigm that extends Cloud computing and services to the edge of the network.

A remarkable step for Fog Computing is the creation of the OpenFog Consortium in 2015. The OpenFog Consortium is a partnership of tech industry and academic institutions to promote the adoption of Fog Computing. To this purpose, it published several technical reports and white papers with the aim of proposing standard for Fog Computing. Later in 2019, the OpenFog Consortium merged with the Industrial Internet Consortium to continue the promotion and the developing of best practices both for Fog and Edge computing. Following the momentum, a wide number of work on Fog Computing were published. The major contributions were well summarized by Mouradian et al. in a comprehensive summary on Fog Computing in [24]. Another notable survey, which describes contributions to different areas of Fog Computing and open research challenges is [28]. A relevant book that presents the contributions and challenges for both Edge and Fog Computing is [29].

As aforementioned along this thesis, Fog and Edge Computing are widely interchanged in literature given their similarities. To this end, dealing with both Fog and Edge Computing is the survey in [7], in which the authors refers to Edge Computing and MEC as Fog related paradigms, which concentrates on the edge side of the network. To outline the difference between the two paradigms, the authors in [30] argue that fog includes "cloud, core, metro, edge, clients, and things" to realize a continuum of computing services from the Cloud to the IoT. This is widely referred as the Cloud-IoT continuum [6, 24]. On the same view is the work in [31],

in which the authors state that Fog Computing jointly works with the Cloud, while Edge Computing excludes the Cloud. In some sense, Fog Computing can be considered a superset of Edge Computing because it contemplates the whole Cloud-IoT continuum into account.

Chapter 3

Concepts

Fog and Edge Computing management is a compelling topic that this thesis aims to contribute to using the novel criterion of Value-of-Information (VoI). To introduce the basic concepts that will be the foundation of the middlewares presented in this thesis, the following Sections illustrate the concept of VoI for Fog and Edge computing.

Then, this chapter present the Adaptive, Information-centric, and value-based (AIV) information maturity model as the guideline on which developing a new generation of applications for Fog and Edge Computing. The AIV maturity model would enable the definition of value-based service components capable of seamlessly adapt their workload to the current execution context. Finally, this chapter provides a formal notation for modeling the allocation of services on fog and edge devices. This notation along with the VoI are the building blocks for developing a new generation of value-based management middlewares.

3.1 Value-of-Information for Fog and Edge Computing

Originally born from the seminal research by Howard in 1966 [16], Value-of-Information (VoI) has long been a widely explored topic in decision sciences. It provides a metric that enables to quantify the value of a single piece of information in decision making. Recently, VoI has started attracting the attention of several researchers in the wireless communications and network and service management research areas as a natural criterion for optimization of (scarce) resource allocation.

Initially, VoI has been applied to military research in [21, 32] to optimize bandwidth consumption and information delivery in tactical networks. In such situations, VoI based information delivering techniques have proven their abilities in dealing with low-bandwidth and disrupted networks. On the other hand, researchers have

started to investigate the application of VoI for different use-case scenarios. In [17], Turgut et al. present the value of information and the cost of privacy for the IoT. A formulation to quantify the value of an information/service that considers the costs (privacy, infrastructure, and maintenance) is given and discussed. Focusing instead on Cloud Computing applications, Boloni et al. propose an interesting VoI-based computation scheduling in [18]. The proposed model considers both the costs and the benefits associated with the processing of information on Cloud Computing platforms and shows how their VoI scheduling approach can outperform other traditional methodologies.

In [19], Giordani et al. discuss the application of VoI to prioritize the most important transmissions in future vehicular networks. The authors quantify VoI utilizing information quality, space, and time inter-dependencies criteria and consider its adoption as proxy for broadcasting decision in vehicular networks. Other works suggest the adoption of VoI to quantify the relevance of information and sensors such as the authors in [20], which propose VoI as a technique to design a sensor service ranking mechanism. Finally, another relevant work considering the possibility of measuring user-utility in the VoI formulation is [22].

In the last few years, researchers have developed service [33] and communication platforms [21] capable of estimating the VoI of information to prioritize its processing and dissemination. A relatively less explored, but possibly even more interesting research avenue is related to the exploitation of VoI as the criterion for optimizing the Fog service fabric. In fact, ranking services and service components according to the total amount of VoI they provide to end users represents a natural and effective approach to realize self-adaptive services for Fog computing applications. By using the amount of VoI delivered to end users as a resource assignment criterion, a system will be able to naturally and seamlessly prioritize the assignment of resources to services that are providing the highest value to their end users - either because they are serving a considerable amount of users or because they are providing highly valuable information.

This thesis considers VoI as an enabling criterion for addressing the challenges of Fog and Edge Computing. In particular, this thesis embraces the idea that developing a service processing model that incorporates the concept of VoI can be an effective criterion to address the processing of the deluge of IoT data.

To fulfill to the absence of the service model leveraging on the VoI concept, the next Section presents an innovative formulation for applying VoI methodologies and tools to Fog and Edge Computing processing models. It is worth noting that, in the context of this thesis, referring to Fog services instead of Edge services is not a lack of generality. On the contrary, this thesis refers to Fog service to indicate that a “general service” can be allocated along the whole Cloud-IoT continuum.

3.2 Service and Information Maturity Model

Adaptive, Information-centric, and Value-based (AIV) proposes an innovative Fog and Edge service model, that enables the development of services capable of automatically scaling their resource requirements to match users' QoE requirements to the best of the system's capabilities given the current execution context. More specifically, AIV assumes that the processing function of a Fog service emerges as the result of the coordinated orchestration of adaptive and composition-friendly service components. 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.).

Developers will define and implement Fog services as a topology of service components that are connected together in a dynamic service fabric, according to a service description that defines the service semantics and characteristics and the interactions between service components. In turn, service components represent the basic building block of Fog services.

AIV also proposes an information maturity model which divides messages in three different categories, according to the corresponding information processing phases. The first processing phase regards *raw data*: input feeds of (typically sensing) data gathered from IoT sensors, smartphones, wearable devices, and other IoT devices in general. According to the AIV model, raw data are analyzed by generic lower level service components to produce higher-order data constructs called *Information Objects (IOs)*. In particular, IOs are generated by multiple aggregated and/or distilled raw data fused 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 worth noting that the generation of an IO could require many raw data objects, and that, in a similar fashion, a CRIO can be the result of the aggregation of multiple IOs provided by many different intermediate service components. Generally, the AIV model considers IOs as the output of service components and CRIOs as the output of Fog services.

3.2.1 Notation

Building upon the AIV model discussed in the previous Section, we introduce the following notation, which we will use as a basis to formulate the optimal service fabric allocation problem. $\mathbb{DS}$ is the list of data sources, which represent IoT sensors or other types of device that generate raw-data to be processed. Each data source

$ds_i \in \mathbb{DS}$ has location ds_i^l.

With regards to processing devices, $\mathbb{D}$ is the list of edge devices, on which it is possible to execute service components that implement a (portion of a) Fog service. Each device $d_i \in \mathbb{D}$ has location d_i^l and total resources d_i^{tr}.

$\mathbb{S}$ is the set of Fog services to allocate. $s_j \in \mathbb{S}$ represents the j-th service. In turn, service s_j is realized as a workflow s_j^{wf} built on top of a set of different service components sc_k, with a corresponding amount of requested resources at time t $sc_k^{rr}(t)$. More specifically, a workflow s_j^{wf} represents an ordered list of service components interacting each other, e.g. $s_j^{wf} = sc_1, sc_2, sc_3, .., sc_n$.

A service s_j is then defined as the combination of its service components $\mathbb{SC}_j$. $\mathbb{SC}$ is the set of service components that need to be allocated. In this work, we assume service isolation to simplify the system model notation, i.e., no service component shared between services:

$$\mathbb{SC} = \bigcup_j \mathbb{SC}_j \tag{3.1}$$

However, by assuming service isolation we are not considering service components shared between different services.

$\mathbb{U}$ is the set of users. Each user $u_l \in \mathbb{U}$ has location u_l^l and it is subscribed to a set of services u_l^s.

It is possible to model the allocation of service components sc_k on edge devices by using matrix notation. $\mathbb{A}$ represents the current service allocation configuration: it is a matrix of size $|\mathbb{SC}| \times |\mathbb{D}|$, where $|\mathbb{SC}|$ is the number of service components and $|\mathbb{D}|$ is the number of edge devices. The matrix $\mathbb{A}$ is composed of $\alpha_{k,i}$ elements: $\alpha_{k,i}$ is equal to 1 if the service component sc_k is allocated on fog device d_i. Therefore, according to this formulation, a service s_j is "fully instantiated" if all its service components $sc_k \in \mathbb{SC}_k$ result allocated at time t. Or, equivalently, if:

$$\sum_{k \in \mathbb{SC}_k, i \in \mathbb{D}} \alpha_{k,i} = |\mathbb{SC}_k|. \tag{3.2}$$

(Note that service components might, and typically will, be allocated on different edge devices.)

The resources allocated at time t on device d_i are $ra_i(t) = \sum_{j \in \mathbb{A}} \alpha_{j,i} \times sc_j^{rr}$, thus leaving $ar_i(t) = 1 - ra_i(t)$ available resources at device d_i and time t. The overall assigned and available resources at time t can then be expressed as $ra(t) = \sum_{j,i \in \mathbb{A}} \alpha_{j,i} \times sc_i^{rr}$ and $ar(t) = 1 - ra(t)$ respectevely.

$$\forall i \quad \sum_{j \in \mathbb{A}} \alpha_{j,i} \times sc_j^{rr}(t) \leq d_i^{tr}. \tag{3.3}$$

Equation (3.3) defines a constraint on the edge resources in use at time t in the overall Fog Computing scenario. In fact, the amount of allocated computational resources allocated by service components running on a device $d_i \in \mathbb{D}$ must be less or equal than amount of total resources d_i^{tr}.

3.3 Managing Fog Computing Service Fabric

Two of the most characterizing aspects of Fog Computing environments are the relative resource scarcity and the considerable workload dynamicity - demanding resource assignment and computing models with fine-grained adaptivity. Developers of Fog services cannot assume to have enough computational, storage, and communication resources to analyze all the incoming data using the full-fledged / sophisticated / fine-grained analytics techniques developed for Cloud environments. Instead, they have to adopt trade-offs: either services discard some data or they have to remodulate analytics so that when they run in the Fog they decrease their computational requirements, perhaps switching to coarser grained but less computationally demanding algorithms.

Aligning to the best practices currently developed in the microservice computing paradigm, many developers typically define and implement Fog and Edge services as a topology of service components that operate relatively independently and are connected together in a dynamic service fabric, according to a service description that defines the service semantics and characteristics, and the interactions between service components.

This raises the opportunity, at the software platform level, to take advantage of the modular architecture and support the dynamic modification of the distributed deployment topology over time by allowing service components to be migrated between Fog devices in an adaptive and context-aware fashion. More in general, Fog services aim to deliver enhanced Quality of Service (QoS) or Quality of Experience (QoE) levels to provide a valuable utility to their end-users [34]. To provide such service levels, they need to be optimal orchestrated and managed on the available Fog resources.

Fog service fabric management, i.e., the dynamic assignment and management of resources in Fog Computing scenarios to fulfill service level objectives and the reweaving of network connections between Fog service components. However, service fabric management in Fog Computing also represents a particularly challenging task, which involves the dynamic management and orchestration of resources to find a suitable allocation for services in a Fog Computing environments.

For instance, Fog scenarios often present heterogeneous computational resources dislocated in several layers: at the edge of the network, in a “fog layer”, or on Cloud

platforms [5]. Moreover, the devices at the edge are characterized in general by limited computational resources that need to be shared among different applications. Other important challenges are related to network management at the edge [35], such as the accurate selection of network links, which should be chosen accordingly to ensure the services' requirements in terms of latency and bandwidth.

To address these requirements, there is the need for service fabric solutions capable of managing multiple Fog services on the available computational nodes. In order to operate an efficient management, service fabric solutions for Fog Computing must collect several information such as the workload of different services, information on the network (latency, available bandwidth), geographical location of devices and their capabilities. Building upon this information, service fabric solutions should be capable of finding the best allocation for a set of services, manage their requirements by scaling the computational resources when needed, and to reactivate or migrate a service when a Fog device becomes unavailable.

To fully realize the resource remodulation potential of the Fog, there is the need to investigate service fabric management frameworks that are capable of efficiently dealing with the ever changing and complex nature of the computing environment. At the same time, there is the need to support those frameworks by considering the adoption of optimization criteria that allow to exploit the limited resources available in the Fog in the most effective way.

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

Chapter 4

Methodologies and Tools

Several tools have been proposed to address the management of Fog and Edge Computing. The majority of them tend to extend Cloud concepts at the edge of the network for allowing the orchestration and management of services on top of IoT gateways and other edge devices. However, few of them consider the idea to integrate VoI methodologies into their design. Therefore, there is the need to investigate the development of new solutions with the twofold objective of realizing i) a functional middleware implementation and ii) tools for evaluating the configuration of service components.

As the basis of comprehensive VoI middleware for managing Fog and Edge Computing, this Chapter presents two valuable tools that will be used for implementing and validate the solutions presented in this thesis. Firstly, this Chapter presents the SPF middleware, a platform for enabling the management of value-based services along the IoT-Cloud Continuum. Then, to allow a realistic evaluation of Fog Computing scenarios this Chapter describes Phileas, a discrete event simulator that aims at enabling the reproducible simulation of value-based Fog services [36]. As SPF, Phileas is built on the top of the AIV model to simulate the value-based processing and dissemination of information in Fog and Edge Computing. Phileas would allow service providers to evaluate different service policies and allocation strategies.

4.1 Modelling 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 [37] which describes a Distributed Dataflow inspired by digital signal processing techniques and allows BPEL-like service definition via scripting solutions. More specifically, 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 re-

quirements of information processing tasks and the strict resource constraints that characterize Fog and Edge 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, 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 VoI metric, which measure the utility that an IO provides to its recipient, represents a very interesting criterion for the purpose of information prioritization.

The AIV information maturity model, described in Section 3.2, is built on the top of these concepts and proposes an adaptive and content-based 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 and edge 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.). Finally, pipelines implements basic processing functionalities that can be easily re-used by different services and pre-installed by Fog providers both in edge/fog devices or in the Cloud. Instead, services that implement application-specific tasks need to be designed and implemented by Fog and Edge application developers.

Believing that this information maturity model maximizes the opportunities to reuse processing components and generated results, the following sections present two different tools: the SPF platform and the Phileas Simulator. The first aims to be reference implementation of the AIV model and it provides a set of function to deal with service and resource management in Fog and Edge Computing. Instead, the second is discrete event simulator for reenacting the same type of services on realistic Fog and Edge Computing scenarios. Both SPF and Phileas will be valuable tools along the following pages of this thesis to present and to evaluate more comprehensive middlewares that adopt an holistic perspective in dealing with the challenges of Fog and Edge Computing.

4.2 Sieve Process and Forward (SPF)

SPF is a Fog-as-a-Service (FaaS) platform [33] for dealing with the IoT information processing at the edge of the network, thus exploiting the proximity of raw-data sources (IoT sensors) and users. SPF identifies different roles for the stakeholders: administrators which are responsible for deploying, running, and operating the SPF components, service providers that develop and deploy IoT applications, and the users of the SPF applications.

SPF is based on the Adaptive, Information-centric, and Value-based (AIV) information model discussed during this Thesis. To deal with the deluge of IoT data, SPF ranks each information object (raw-data, IO, CRIO) according to its VoI value to prioritize the processing and dissemination of critical information at the edge of the network.

To provide readers an architectural overview of the SPF platform, Fig. 4.1 illustrates the two main components of the SPF architecture: the SPF Controller and the SPF Programmable Internet Gateways (PIGs). The SPF Controller acts as an interface between the users and multiple SPF PIGs. In addition, the SPF Controller is responsible for dispatching the requests for services received from the users to the available PIGs, which will execute the required services.

Within the SPF architecture, management functionalities are provided by the SPF Controller, which is responsible for deploying the information processing and dissemination functions required by the registered applications. Using the management functionalities, the SPF Controller can also reprogram the PIGs in the case a new service is required using a proprietary programming interface. The Information Processing and Dissemination functions are instead provided by Proactive Dissemination Service (DSPro) [21], which leverages the set of filtering and communications functions implemented by the software platform, according to the commands received by the SPF Controller. DSPro is part of the Agile Computing Middleware

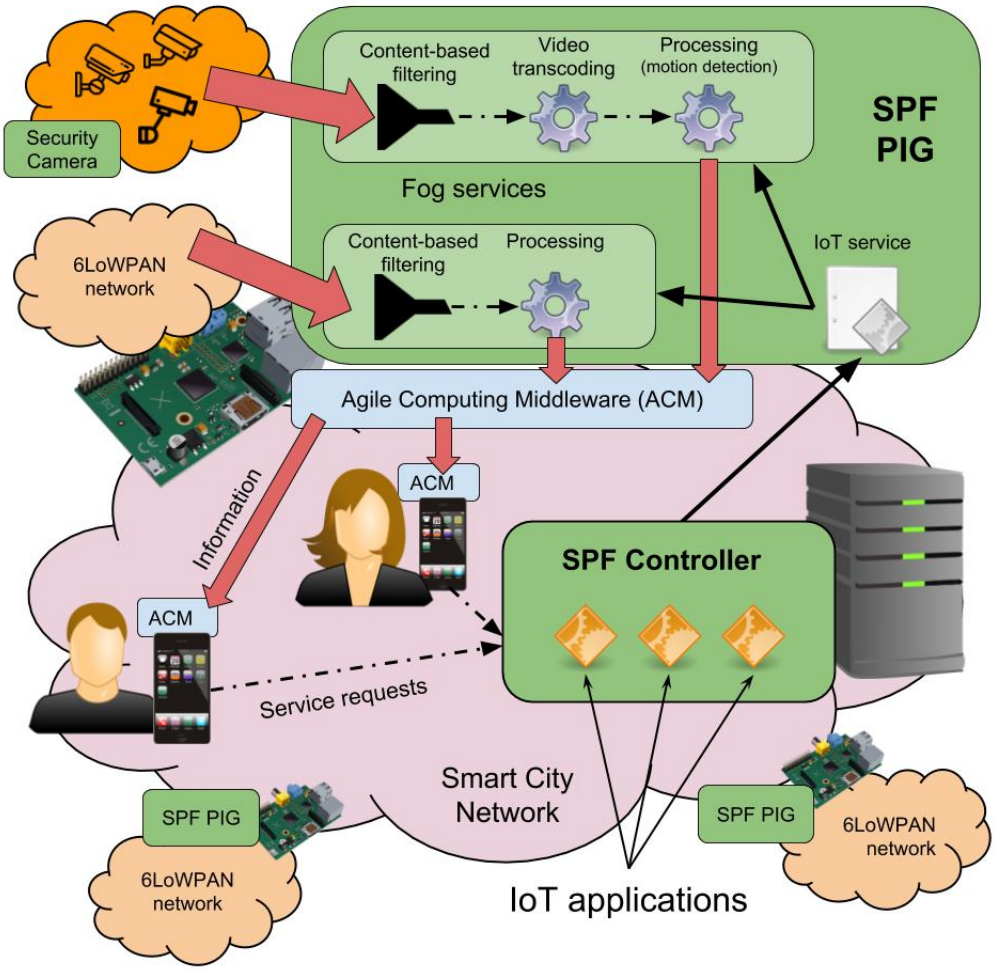


Figure 4.1: SPF in a Smart City Scenario

(ACM), a more comprehensive middleware for addressing communications in harsh and challenging environments [38].

With regards to the location of PIGs, they can be deployed directly on gateway nodes at the edge of the network on dedicated hardware placed in the gateway nodes' proximity (as illustrated in Fig. 4.1). Another option is to deploy PIGs on the Cloud to run services requiring high computation requirements. On the one hand, PIGs located in the proximity of IoT assets can provide reduced communication latency at the price of minor computational resources. On the other hand, PIGs deployed on the Cloud can benefit of scalable computational resources with higher communication latency. This makes SPF capable of allocating computational tasks along the whole Cloud-IoT continuum, thus providing a wider range of options to developers and administrators of where deploy services. Therefore, it is in the responsibility of developers and administrators to choose the configuration which best fit their requirements.

With respect to SPF applications, they are available to users that can request the execution of services to the SPF Controller over the available communication

links, e.g. WiFi, LTE. When the SPF Controller receives a request for a service, it analyses the information contained in the request to select a PIG where to execute the service. The selection process is based on several criteria such as the proximity to assets and the available PIG computation and network resources. For example, if a user requests to run a face recognition algorithm on a specific camera, the SPF controller will search for a PIG in close proximity of the camera asset. In turn, the PIG will start executing the face recognition algorithm by analyzing the video feed from the requested camera to return to the user the results of the analysis.

Finally, Fig. 4.1 shows an illustrative example of SPF operating in a Smart City scenario, in which IoT services provide useful information to smart citizens and other Smart City operators. In this example, raw-data sources, such as traffic and security cameras, usually dedicated to day-to-day monitoring of the city, capture data feeds that will be consumed by IoT services running on top of SPF PIGs. These services analyze the raw-data to produce CRIOs into one or multiple service processing phases depending on the service's characteristics. A multi-phase processing example is the motion detection service illustrated in Fig. 4.1, which leverages on a pre-processing service component that runs a video-transcoding algorithm on the camera feeds before passing them to the motion detection service component. Finally, the motion detection service component elaborates the transcoded video feed to identify motion activities and to deliver this information to the end-users who requested it.

4.3 The Phileas Simulator

Phileas is explicitly developed to allow the reproducible evaluation of Fog and Edge applications in a realistic environment. Phileas allows users to define value-based Fog services built on the top of the AIV model and to test their performance from a VoI perspective. Phileas is implemented in Ruby and distributed as open source (MIT license) on GitHub at the `https://github.com/DSG-UniFE/phileas` URL.

4.4 State-of-the-art in Fog Simulation

Fog computing applications become more and more pervasive 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. With this regards, simulation is a promising practice in this field of research, since it enables to represent scenarios involving a large number of variables.

Margariti et al. wrote a comprehensive survey on Fog Computing simulators in [39]. In this article, the authors analyze several Fog Computing simulators from a cost perspective and discuss their models and limitations.

A relevant simulator is iFogSim [40], which aims to offer a complete toolkit for modeling the allocation of tasks and resources in Edge and Fog Computing environments. A similar work is EdgeCloudSim, which was developed for providing a deep introspection on MEC environments [41]. Furthermore, EdgeCloudSim provides features for simulating: dynamic load of requests, devices and users mobility, networking, and service orchestration. Both iFogSim and EdgeCloudSim take inspiration from CloudSim [42], a simulator for Cloud Computing environments.

Another effort in the same direction is the one the described in [43], where the authors present a simulator based on Discrete EVent System Specification (DEVS) 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 on the number of the users and the services' allocation between the Fog and the Cloud. Finally, the YAFS simulator [44] aims to be a comprehensive tool for modeling applications and network in dynamic Fog Computing scenarios.

4.4.1 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 [45, 46].

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 to build 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. Future versions of the simulator 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 behind the development of Phileas.

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 to define the share of users within the group interested to a specific content type.

4.4.2 VoI Tracking and Processing

Phileas accurately models the VoI of each message, from its generation to its consumption. More specifically, in Phileas 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.

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 from the VoI of messages that need to be processed to generate that message according to service-specific policies.

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 of a message as time passes and the information travels from its originating source, according to the configuration of the entity that generated the message (data source or service component).

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

4.4.3 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 be stochastically modeled according to a Gaussian mixture distribution. This is a realistic approach, as proved by our recent research [47]. 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 NS3 [48], whose 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.

Phileas models 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 [49] [50] [51].

4.4.4 Mobility Models

The realistic evaluation of Fog Computing scenarios should support different mobility models for the entities involved in the simulation. This is particularly interesting for Smart City scenarios where users can move to several locations during the simulation time.

Mobility models for wireless and ad-hoc networks have been widely investigating in literature [52]. They make possible to evaluate the performance of protocols or configurations when nodes move through a defined space. For example, one of the first and a very simplistic mobility model is Random Walk, in which nodes follow unpredictable and erratic movements of entities. Another one is the Random Waypoint mobility model, which extends Random Walk by enabling the description of waypoints, where moving entities can stop before changing directions. Others and more complex mobility model exist to fulfill the needs of specific environments.

It is worth noting that the majority of mobility models implemented for simulators rely on 2-dimensional and 3-dimensional grids, on which is possible to calculate the positions of entities given their velocity and direction. However, Phileas adopts a more realistic modeling of locations, where each location object has an associated longitude and latitude in accordance with the GPS. This makes the adoption of mobility models more challenging because it requires to calculate new positions in a realistic fashion.

Considering these constraints, Phileas provides as built-in the Random Walk mobility model with different configurations that let users describe geographic boundaries, starting and ending coordinates, and directions. Furthermore, Phileas also allows describing the movements of flying objects such as helicopters and UAVs that interact with the simulation. Finally, it is possible to extend the range of implemented mobility models by extending the mobility model class with the behavior of the desired mobility model.

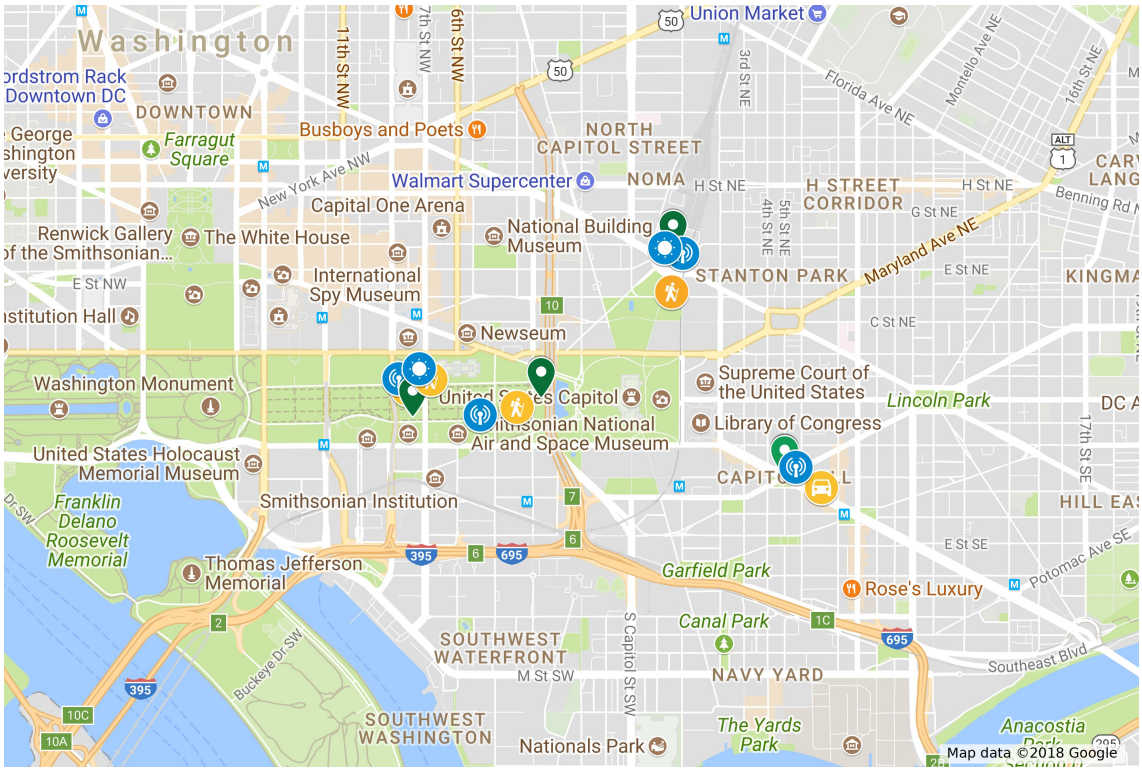


Figure 4.2: Location of the data sources and service components for the fictional Fog Computing scenario considered in the experimental evaluation.

4.5 Experimental Evaluation

In order to evaluate the Phileas effectiveness and potential in analyzing the performance of Fog applications based on the AIV model, the following experiments consider a fictional Fog Computing application running in Washington DC, USA. As illustrated in Fig. 4.2, the fictional application offers different typical services of a smart city scenario. In particular, the scenario is set in the downtown area of Washington DC to setup different data sources, which will feed pollution, weather, music, and traffic monitoring services running on several edge devices (green icons in the picture) and on the Cloud. Finally, the scenario defines multiple user groups 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. The music recognition service analyzes sound's samples gathered by sensors located in the park to recognize, using OCR capabilities, the corresponding track's name and author. This service requires high computational and data resources and it is intended to be executed on the Cloud. The weather service is running in multiple locations and collects humidity, temperature, and pressure data from weather's sensors to produce forecast information. Finally, the traffic service gathers data from the cameras located at street

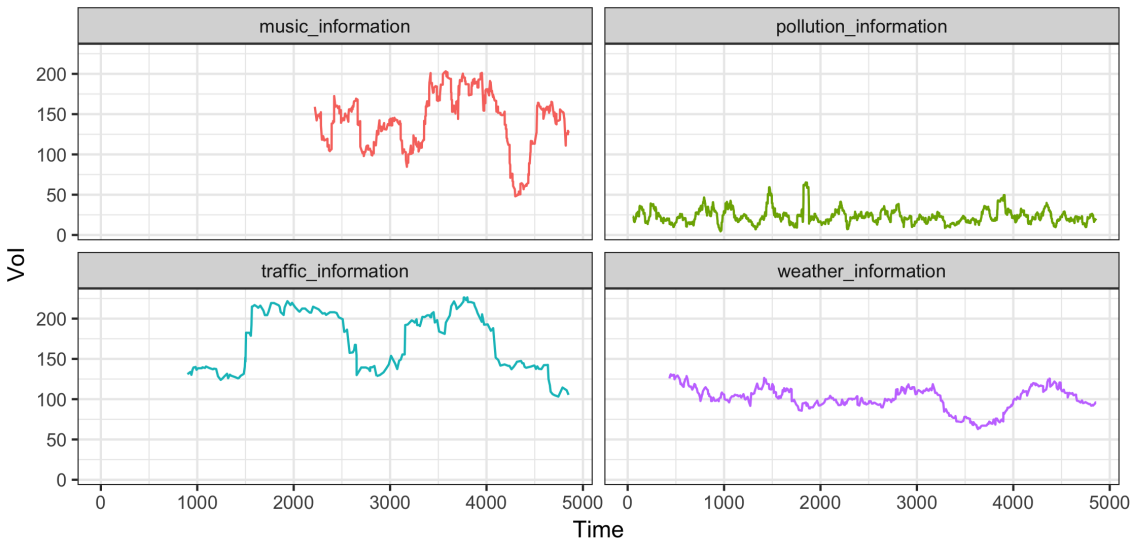


Figure 4.3: Simulated VoI trend during simulation time for defined services.

intersections to determine the status of the viability and inform the users interested in it.

For the purpose of this experiment, the information processing logic of the services is built on the top of 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 $p = 0.9$. 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.

User groups are distributed among the scenario's area and mainly depending on their position, have different share interests in receiving the information offered by the different services. For instance, the 80% of the people walking in the National Mall park are interest in receiving notification about quality of the air or the events taking place in the park, e.g. if a concert is being played in the area. In addition, the 85% of the users that are driving vehicles are interested in receiving traffic monitoring information to adapt their route accordingly. Note that, as explained in the previous Section, the number of the users interested in a service contributes to the VoI calculation of the output CRIOs, and thus affects the tasks allocation and processing strategies of the applications running on the available devices.

To collect the experimental data, the simulation duration is set to one day. Fig.

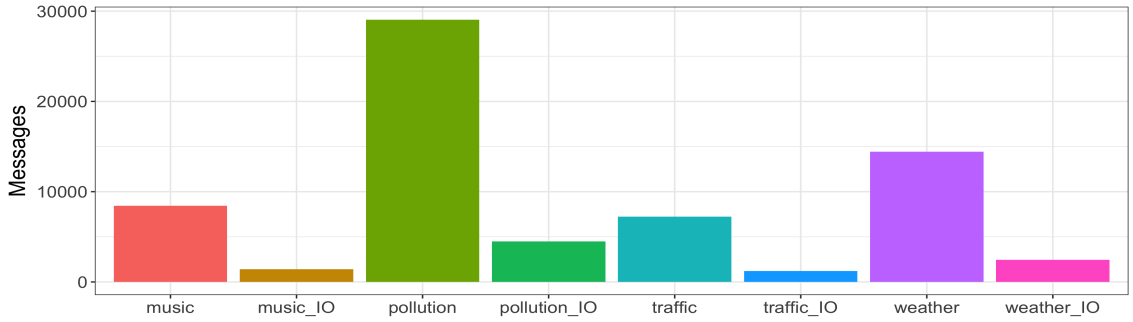


Figure 4.4: Number of dropped messages (raw-data and IOs) for service components.

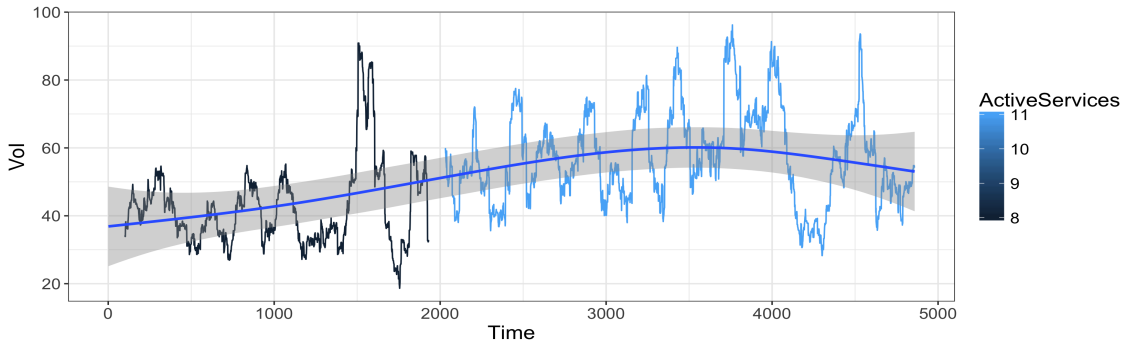


Figure 4.5: Simulated VoI as a function of the number of running service components.

4.3 illustrates the total amount of Value-of-Information recorded for the CRIOs produced by each different service: music, pollution, traffic, and weather information. As expected, some services have higher VoI than others, in particular the music and traffic service, which per definition offer real-time/time-sensitive information. Furthermore, Fig. 4.3 also illustrates how the total VoI remains high even when the services allocated saturate the entire amount of resources available. More specifically, the activation of the music service ten hours after the begin of the simulation does not affect significantly the VoI information of the other services.

Another interesting result is to observe 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 4.4 shows how the percentage of dropped messages changes in function of the service. As 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. 4.4 with Fig. 4.3 one can notice that the VoI information is not affected by the dropping of the messages, since its value remains high.

Finally, Fig. 4.5 illustrates the total amount of VoI generated in the fictional scenario with a curve color-coded according to the number of running services. 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, with higher-VoI messages are prioritized over lower-VoI ones, and thus leading to the VoI conserving trend shown in Fig. 4.5.

4.6 Configuration

As final note, this Section provides an example configuration for describing a Fog Computing Scenario. This is to highlight the highly descriptive interface that Phileas provides to describe all components involved in the simulation of a Fog Computing scenario:

Example configuration for locations:

```
locations(
  #national mall edge device
  1 => { latitude: 38.889809, longitude: -77.014666 },
  #national mall pollution and weather sensor data_source
  2 => { latitude: 38.889070, longitude: -77.018466},
  )
```

Example configuration of a data source (IoT sensors):

```
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_information,
    output_message_type: :crio,
    resource_requirements: 50.0,
    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: 2.0,
    },
  },
)
```

Example of service activation:

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

Example of 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: :music_information,  share: 0.2 },
      { content_type: :weather, share: 0.7 },
    ]
  }
  )
```

Chapter 5

VoI-optimal placement of processing tasks

After having presented methodologies and tools that integrate the VoI concept, this Chapter presents the VoI-optimal placement of processing tasks in Fog and Edge Computing. More specifically, this chapter analyzes both user-agnostic and user-specific VoI models for proposing a management framework that addresses the optimal placement of service components from a value-based perspective [53].

To achieve this goal, this Chapter presents an innovative user-specific VoI model, which extends the state-of-the-art in VoI research by investigating the VoI utility for each user on a per-message basis. This model is the basis to implement the VoI optimization of service components in Fog Computing. Furthermore, to investigate others simplified optimization solutions, this Chapter proposes a distance based heuristic for approximation and other model refinements which provide very promising results. This work aims to develop valuable techniques that can be integrated into management middlewares to select the best allocation for processing tasks among a pool of available devices.

5.1 Optimal resource allocation problem

Building upon the AIV model discussed in Chapter 3 is possible to formalize a management framework for the allocation of Fog service components. In fact, the problem of resource allocation for service fabric in Fog Computing applications requires a mathematical optimization framework. More specifically, this problem aims to maximize a function f_n that captures the behavior of the system according to the optimization criterion in the n-th window of time (which goes from t_n to t_{n+1}):

$$\begin{aligned} \underset{\mathbb{A},\,\Gamma}{\arg\max} \quad & f_n(\mathbb{A},\Gamma,\mathbb{D},\mathbb{S},\mathbb{U}) \\ \text{s.t.} \quad & \forall k \sum_{m\in\mathbb{SC}_k, i\in\mathbb{D}} \alpha_{m,i} = |\mathbb{SC}_k| \end{aligned} \tag{5.1}$$

It is worth noting that function f_n changes every time the global system state $\Theta = (\mathbb{D},\mathbb{S},\mathbb{U})$ changes, e.g., when new resources become available or new service components are instantiated. In that case we have a new function to optimize, i.e., f_{n+1}.

Using a sequence of functions to model the system implies that we are capable of finding out when the function changes, i.e., when the system changes – which is a quite complex task per se (see for instance [54], [55], [56], and [57]).

Since it is going to be difficult to detect (or predict) changes to the global system state, and consequently to find out when the optimization procedure should be re-triggered, it might be convenient to reformulate the optimization problem in equation (5.1), switching from a model based on a sequence of static (i.e., time-invariant) optimization functions f_n to one based on a single dynamic (i.e., time-varying) optimization function f:

$$\begin{aligned} \underset{\alpha,\,\gamma}{\arg\max} \quad & f(\mathbb{A},\Gamma,\mathbb{D},\mathbb{S},\mathbb{U},t) \\ \text{s.t.} \quad & \forall k \sum_{m\in\mathbb{SC}_k, i\in\mathbb{D}} \alpha_{m,i} = |\mathbb{SC}_k| \end{aligned} \tag{5.2}$$

This turns the optimal resource allocation problem from a sequence of (relatively) static optimization problems to a single – but significantly more complicated – dynamic optimization problem.

5.2 VoI Optimization

In Fog and Edge Computing, a particularly attractive metric to consider for the adoption within a value-based management framework is the total amount of VoI delivered to end users by a given service fabric configuration. From the mathematical formulation perspective, this could be expressed as:

$$f_n = \textsc{Total}_{VoI}(t_n, t_{n+1}) = \sum_{m\in\mathbb{M}(t_n,t_{n+1})} VoI_\Theta(m). \tag{5.3}$$

where $\mathbb{M}(t_n, t_{n+1})$ are the messages received by end users within the (t_n, t_{n+1}) time window.

Note that for the purposes of the VoI maximization criterion the VoI contribution of messages exchanged between sensors and service components are considered

indirectly in the calculation. In fact, the starting VoI of a CRIO message is the result of the aggregation of the multiple raw-data and IOs who generated it. Also note that a message can be received by different end users.

Of course, the adoption of VoI-based Fog service fabric management requires the development of accurate, generically applicable, and practical models to estimate the VoI delivered by IOs. Devising VoI models that are cabaple of tracking the changes of VoI in IOs while retaining a reasonable complexity is quite a challenging task because, as the formulation $VoI_{\Theta}(m)$ highlights, the VoI associated to a message m actually depends on the current state Θ of the system. The next Section presents some proposals to define the VoI_{Θ} model, adopting different perspectives and assumptions on the knowledge of the system.

Table 5.1: Summary of used notation

Term	Meaning
$\mathbb{DS}$	Set of data sources
$ds_i \in \mathbb{DS}$	i-th data source
ds_i^l	Location of i-th data source
$\mathbb{D}$	Set of edge devices
$d_i \in \mathbb{D}$	i-th device
d_i^l	Location of i-th device
d_i^{tr}	Total resources at i-th device
$\mathbb{S}$	Set of Fog services
$s_j \in \mathbb{S}$	j-th service
s_j^{wf}	Workflow of j-th service
sc_k	k-th service component
$sc_k^{rr}(t)$	Amount of resources requested by k-th service component at time t
$\alpha_{k,i} \in \mathbb{A}$	Service component sc_k allocated on device d_i
$ra_i(t)$	Amount of resource allocated on device d_i
$ar_i(t)$	Amount of available resources on device d_i
$\mathbb{U}$	Set of end users
$u_l \in \mathbb{U}$	l-th end user
u_l^l	Location of l-th end user
u_l^s	Set of services the l-th end user subscribed to
r^t	Generation time of request r
r^l	Location of request r
m^{ot}	Originating time of message m
m^{ol}	Originating location of message m

5.3 VoI Models

Most VoI models proposed so far in literature only consider the (simpler) case in which the VoI of a message does not depend from the specific user that receives it. These approaches implicitly assume an "objective VoI perspective", according to which the value of an information object is an intrinsic characteristic and does not depend from the user consuming it. More sophisticated approaches based on a "subjective VoI perspective" require a deep knowledge and differentiation for each possible user, thus further increasing the complexity of the model. This Section attempts to present generically applicable models of both types.

5.3.1 User-agnostic VoI Model

At the moment of its generation each message (of either raw data, IO and CRIO type) has an originating VoI. The starting VoI of IOs and CRIOs depends from the message- and service component-specific result of the processing of previous messages. The starting VoI associated to each IO is calculated as a service component-specific function of the raw data messages consumed in the generation of the IO.

The starting VoI associated to each CRIO can be calculated as a function of the priority of the corresponding service and of the output of a service-specific VoI calculation function, that every service implementation is supposed to provide:

$$VoI_0(m, \mathbb{I}_m, s) = SSV(\mathbb{I}(m)) \times FSP(s)) \tag{5.4}$$

where $\mathbb{I}_m$ is the set of input IOs processed in the generation of CRIO m, s is the Fog service that generated m, and $SSV(\mathbb{I}(m))$, as in "Service-specific Value (calculation)", is a factor that allows to take into account service-specific considerations when assessing the value of the information extracted from the $\mathbb{I}(m)$ IOs. Finally, $FSP(s)$, as in "Fog Service Priority", is the priority of service s, thus representing a weight factor that permits to assign higher VoIs to the CRIOs produced by higher priority Fog services.

The starting VoI in equation (5.4) is the basis to calculate the VoI of message m at its recipients for a service s. More specifically, the total VoI delivered by a CRIO m needs to consider factors such as the number of its recipients, timeliness relevance decay (a factor taking into account the decrease in the value for time sensitive IOs by applying a penalty according to the delivery time), proximity relevance decay (a factor taking into account the decrease in the value for location-aware IOs by applying a penalty according to the delivery location):

$$VoI_{\Theta}(m, \mathbb{I}_m, \mathbb{MR}(m), s) = VoI_0(m, \mathbb{I}_m, s) \times \sum_{r \in \mathbb{MR}(m)} [TRD(r^t, m^{ot}) \times PRD(r^l, m^{ol})]$$

where $\mathbb{MR}$ is the (set of) receivals for message m, $TRD(r^t, m^{ot})$, as in "Timeliness Relevance (of Request) Decay", is a factor that takes into account the loss of VoI in the time passed between request generation time r^t, and response generation (the current time m^{ot}). Finally, $PRD(r^l, m^{ol})$, as in "Proximity Relevance (of Request) Decay", is similar to the $TRD(r^t, m^{ot})$ component, but its impact on the final VoI score depends on the physical distance between the position of the requester, given by r^l, and that of the source of IO, provided by m^{ol} (in case message m was simultaneously issued by several requesters, r^l returns the location of the requester closest to the IO source). With regards to the general case in which an IO m is constructed from more raw-data messages, the centroid of the origin points of each raw-data that contributed to its generation is considered for the purpose of PRD calculation. This also applies to CRIO messages resulting from the composition of multiple IOs.

5.3.2 User-specific VoI Model

To account for user specific utilities we could add to equation (5.5) a component (U) that considers utility for each user receiving message m:

$$\begin{aligned} VoI_{\Theta}(m, \mathbb{I}_m, s, r, t) = \quad & VoI_0(m, \mathbb{I}_m, s) \quad \times \\ & \sum_{r \in \mathbb{MR}(m)} [TRD(t, m^{ot}) \times PRD(r^l, m^{ol}) \times U(r^u, m, t)] \end{aligned} \tag{5.5}$$

where $U(r^u, m, t)$ is the utility of message m for user r^u at time t.

Note that this model does not specify the behavior of utility function U, which is likely to be user-specific and context dependent. Most importantly, forecasting the U component in equation (5.5) represents a **challenging task**, even more than the forecasting of the sum component in equation (5.5).

The practical adoption of a high complexity model such as this requires the adoption of *simulation-based optimization*. Simulation-based optimization [58] is based on the creation of a simulation-based model and the optimization of its input parameters. Therefore, it is possible to verify how different service component allocations perform within a defined scenario by evaluating the total VoI collected within the given time window.

Model Simplifications

It is worth to formulate a simplification of the model defined in equation (5.5) to reduce its complexity to a manageable degree in order to avoid simulation based approaches for its evaluation. More specifically, it is possible to assume that VoI_0 only depends on message type (m_t) and service profile:

$$VoI_0(m, \mathbb{I}_m, s) \approx VoI(m_t, s) \tag{5.6}$$

In addition, the sum component in equation (5.5) can be approximated by clustering messages and users in a limited number of classes, calculating an average TRD and PRD for each class of messages, and calculating an average utility per message and user type. In particular, both PRD and TRD functions assumes values in $[0, 1] \in \mathbb{R}$ and they act as decay multiplier. Following this approach, the equation becomes:

$$\begin{aligned} &\sum_{r \in \mathbb{MR}(m)} TRD(t, m^{ot}) \times PRD(r^l, m^{ol}) \times U(r^u, m, t) \approx \\ &\sum_{m\_t \in M} [\overline{TRD(m\_t)} \times \overline{PRD(m\_t)} \times \sum_{u\_t \in U} \overline{U(u\_t, m\_t)}] \end{aligned} \tag{5.7}$$

where m_t represent a class of messages.

With regard to message clustering, it is possible to adopt different policies. For instance, a natural way of clustering messages is according to their content or service type. User clustering could be performed according to several criteria, including location, profile type, and so on. For location-based user clustering, an interesting idea is to model the number of users at a given time t thought the concept of user group. A user group u_k represents a set of users in a defined location u_k^l with share of interest $w_{sj}(k)$ in consuming CRIOs produced by a service s_j. At a given time t, the number of users within a user group u_k is fixed and it is expressed as $u_k(t)$.

We then arrive to the *simplified user-specific VoI model*:

$$\begin{aligned} VoI_\Theta(m, \mathbb{I}_m, s, r, t) = \quad & VoI(m_t, s) \quad \times \\ & \sum_{m_t \in M} [\overline{TRD(m_t)} \times \overline{PRD(m_t)} \times \sum_{u_t \in U} \overline{U(u_t, m_t)}] \end{aligned} \tag{5.8}$$

However, even if simplified, this model still presents some of the disadvantage of the user-specific VoI model in eq. (5.7). More specifically, the evaluation of the PRD and TRD components of eq. (5.8) is still challenging as their values are IO specific. In addition, service components defining the workflow of a service, are likely to be instantiated on different edge devices. Therefore, the calculation of each PRD and TRD requires to consider the location of the edge device where the service

component sc_k is instantiated on. As a result, solving the model defined in equation (5.8) is still challenging and it requires simulation-based optimization techniques, thus hindering the practical adoption of this model.

5.3.3 A distance based heuristic for Service Fabric Management

To address the practicality issues of the models presented above, another possibility is to devise a tractable approximated *distance minimization model* that aims to find optimal allocation for service components on fog devices by minimizing the overall distance defined by the service components allocation. In fact, minimizing the overall distance would minimize the impact of the PRD decay function, thus maximizing the VoI of each message m.

Inspired by the processing stages defined in the AIV information maturity model (raw-data collection, IOs processing, CRIOs delivering), it is possible to devise a Distance Component (DC) for each processing stage to express respectively: proximity to data sources and devices (raw-data), proximity to devices running service components belonging to the same service (IOs), and proximity of edge devices and users (CRIOs) to express the distance minimization model:

$$\min_{\mathbb{DS},\mathbb{D},\mathbb{U}} (\omega_{rd} \times DF_{rd} + \omega_{ios} \times DC_{ios} + \omega_{crios} \times DC_{crios}) \tag{5.9}$$

where:

$$DC_{rd} = \sum_{sc_i \in SC} [distance(d_i^l, ds_i^l)]$$
$$DC_{ios} = \sum_{sc_i \in SC} [distance(d_i^l, dj_i^l)]$$
$$DC_{crios} = \sum_{sc_i \in SC} [distance(d_i^l, u_k^l)]$$
$$\omega_{rd} + \omega_{ios} + \omega_{crios} = 1$$

In these equations, DC_{rd}, as in "DC for raw-data", takes into account the decay the information is subjected to from the generation of raw-data to its first processing phase, i.e. the distance between the sensors generating raw-data and the devices on which service components process the raw-data. DC_{ios}, as in "DC for IOs", is to consider the information-delay in the processing chain, i.e. related service components should be allocated on devices close to each other to obtain reduced distance and latency. DC_{crios}, as in "DC for CRIOS", considers the proximity of the device generating the information in its final stage (CRIOs) to the users interested in consuming it. Finally, the weights ($\omega_{rd}, \omega_{io}, \omega_{crios}$ in (5.9) are to balance the importance of each DC component in a weighted sum method fashion. The

component weights are strictly dependent on the scenario conditions (location of data sources, location of devices, user group locations) and need to be selected empirically. The service component allocations resulting from the optimization of (5.9) will privilege lower distance service component allocations, thus minimizing the delay within the overall service component workflow.

Given the minor complexity, the distance-minimization model (5.9) is well suited for online optimization. To this end, Alg. 1 presents a distance based heuristic. Alg. 1 implements the distance based heuristic using a function that evaluates a given service component allocation, checks for its feasibility, and returns a score value representing the summation of DC components. Furthermore, to balance the importance of each DC component and the priority of service components, the distance based heuristic relies on both the $\omega_{rd}, \omega_{io}, \omega_{crios}$ weights defined in (5.9) and on the service priority weights FSP defined in (5.4).

Delving into details, Alg. 1 evaluates the feasibility of an input allocation considering both the constraints on resources defined in equation (3.3) and a given constraint on a maximum distance, thus avoiding to select infeasible allocations. Then Alg. 1 calculates the DC impact for each service component discriminating between service components processing raw-data, IO, and CRIO messages. Firstly, for each service component processing raw-data messages, the algorithm checks the distance from the device where the service component is allocated to the data sources where those messages are generated. Secondly, for each service component processing IOs instead, Alg. 1 checks the distance with the devices where other service components belonging to the same service are allocated. Finally, for service component processing CRIOs, the algorithms evaluates the position of the device processing that service component with respect to the user groups' locations.

As a final consideration, a different approach to a distance based heuristic is to focus on latency minimization. However, if a strict latency model for edge-to-edge communication is not defined, the latency could be approximated as a logarithmic function of distance. This is a common practice adopted by other network simulators such as NS3 [59] and it well suits wireless end-to-end communications. A reasonable approximation is to relate latency to distance, thus allowing to solve more general Fog scenarios where a latency model is not well defined. To this end, it is worth considering that, this approach is common to other related works such as [60, 61].

However, in those scenarios where a latency model is defined and service components mandate strict latency requirements a latency minimization approach is preferable. To implement such approach, it is possible to substitute the DC components with their latency counterparts, using an empirical latency model or a network simulator to calculate the latency within the scenario.

Algorithm 1: distance based heuristic algorithm

Data: An allocation array x
Result: Sum of the DC components associated to allocation x
initialize DC_sum to 0;
`// check the for each device the constraint on resources`
forall *d in Device* **do**
    assigned_resources = check_resources(d);
    if *unfeasible(assigned_resources)* **then return** ∞;
`// for each service component processing raw-data calculate the distance from device`
`// to the raw-data sources`
DC_rd = 0;
forall *sc in sc_raw_data* **do**
    dc_rd_sc = calculate distance with raw-data sources;
    if *unfeasible(df_rd_sc)* **then return** ∞ ;
    DC_RDdc_rd_sc;
`// for related service component processing IOs calculate the distance from`
`// the devices where they are allocated on`
DC_ios = 0;
forall *sc in sc_ios* **do**
    dc_ios_sc = calculate intra-chain distance ;
    if *unfeasible(dc_ios_sc)* **then return** ∞;
    `// multiply this component for its priority factor`
    dc_ios_sc = dc_ios_sc $\times$ FSP(sc);
    DC_iosdc_ios_sc;
`// for service component processing CRIOs calculate the distance from the device`
`// to the user groups' locations`
DC_crios = 0;
forall *sc in sc_crios* **do**
    dc_crios_sc = calculate distance with user groups;
    if *unfeasible(dc_crios)* **then return** ∞;
    `// multiply sc for its priority factor`
    dc_crios_sc = dc$_c$*rios_sc*$\times$ FSP(sc);
    DC_criosdc_crios_sc;
return $\omega_{rd} \times DC_{rd} + \omega_i os \times DC_{ios} + \omega_{crios} \times DC_{crios}$;

Table 5.2: Distance between user groups and fog devices.

	u_1	u_2	u_3	u_4
d_1	723.55	1586.61	0	339.74
d_2	158.90	2240.61	702.95	366.34
d_3	414.92	1884.99	339.74	0
d_4	0	2297.23	723.55	414.92
d_5	2297.23	0	1586.61	1884.99
d_6	2091.34	206.00	1380.80	1679.33
d_7	136.19	2189.24	606.02	327.48

5.4 Experimental Evaluation

To validate the proposed models, the following experiments rely on a testbed simulating a realistic Fog Computing application scenario similar to the one illustrated in Chapter 4. For calculating the VoI generated by the applications, these experiments leverage the simplified user-specific model in equation (5.8) and the distance based heuristic defined in Alg. 1. Finally, the following experiments use meta-heuristics optimization to identify the configuration for the service fabric of the Fog Computing applications that maximized the total VoI delivered to end users.

This testbed contains the description of 7 devices, 9 data sources, 4 user groups, and 8 service components. Within the scenario, each Fog Computing service is defined as a workflow of 2 service components. The locations of devices, data sources, and user groups are defined in the scenario with a latitude and longitude position in accordance with the GPS position system. In particular, the distance between user groups and devices is depicted in Table 5.2.

The Phileas simulator reenacts the above descripted scenario for the purpose of validation [62]. Phileas was specifically designed to reenact Fog services adopting the AIV service model and implements advanced VoI calculation features that allow to define the VoI model to consider for an application, such as the simplified user-specific VoI model defined in eq. (5.8).

Phileas takes as input a configuration describing devices, users, service of a realistic Fog Computing and an allocation array describing the allocation of service components on devices to return as output the total VoI collected in a simulation time window of one hour.

In the experiments, most of the input configuration remain fixed, while the input allocation array, i.e., the $\mathbb{A}$ parameter of the management framework described in eqs. (5.1) and (5.2), is controlled by the optimization solution that explores the search space in the attempt to maximize the total VoI delivered to end users. To guarantee the reproducibility of the simulation, all the random variables responsible for the event generation have a fixed seed.

5.4.1 Optimization Methods

As the optimization method, the following experiments consider meta-heuristics solutions, which represent an interesting and valuable technique to solve dynamic optimization problems in large and complex search spaces, such as the VoI-based service placement optimization one. More specifically, this framework relies on Genetic Algorithms (GA) [63] and Quantum Particle Swarm Optimization (QPSO) [64], which have proved to be well suited for resource management applications of similar complexity. The implementations of GA and QPSO in the ruby-mhl (`https://github.com/mtortonesi/ruby-mhl`) optimization library.

With regards to the configurations, GA uses a population size of 128 individuals, an integer-based representation, a mutation operator applying random displacements sampled from a discrete geometric probability distribution, and a crossover operator based on a variant of extended line recombination algorithm designed to operate on integers. In addition, Fig. 5.1 depicts the representation used for encoding the service component allocation on devices. In this representation, each block represents the mapping of a service components on a fog device. In other words, the value of a block indicates the device where that service component will be allocated. More specifically, Fig. 5.1 identifies each service component sc_k as part of service s_j using the notation $sc_{k,j}$, where k identifies the component within the workflow of service s_j. In the representation, service components belonging to the same workflow are adjacent blocks.

In turn, the QPSO algorithm uses a similar representation for the search space. As is often proposed when adopting PSO-based algorithms for integer or mixed-integer optimization problems, these experiments considered an $\mathbb{R}_n^+$ search space and applied rounding to the next integer on the output of the algorithm, i.e., the best particle position. In addition, given the problem size and complexity QPSO is configured to use a 40-particle swarm and a contraction-expansion coefficient $\alpha = 0.75$.

Finally, it is worth noting that these representations implicitly solve the constraint defined in equation (3.2), by allocating each service component to a Fog device. As a result, it is particularly well suited for the service placement optimization problem.

5.4.2 Results

The first experiment had the objective of evaluating the performance of the meta-heuristics on the simplified user-specific VoI model. Both GA and QPSO are configured to find the best allocation for service components on fog devices over 100 iterations. Fig. 5.2 depicts how the two meta-heuristics perform in optimizing the

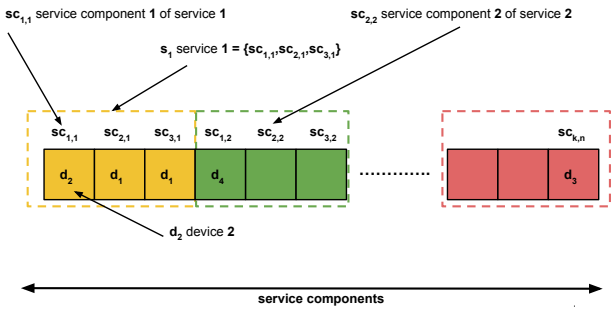


Figure 5.1: Service component representation. Each block represents the mapping between a service component and a fog device.

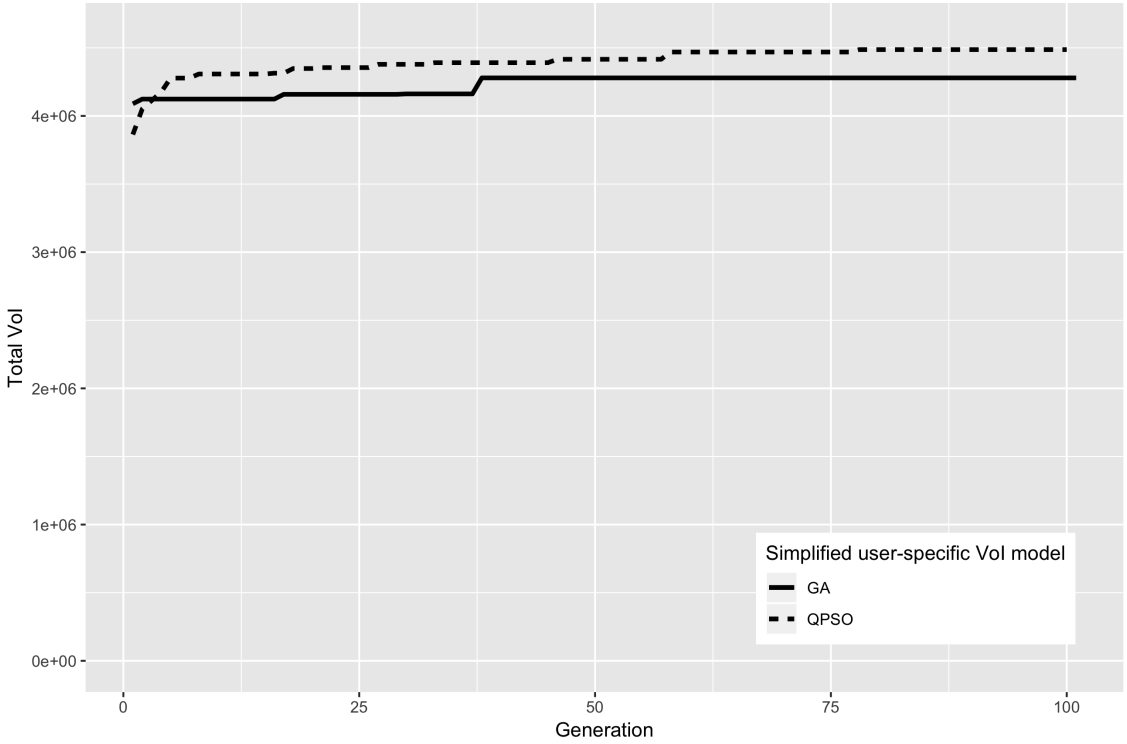


Figure 5.2: Total VoI for each iteration of the two meta-heuristics on the simplified user-specific VoI model.

total VoI in the given time-window. Both GA and PSO converge quickly to a good solution, proving their effectiveness, with GA demonstrating better initial convergence speed and QPSO exhibiting a more explorative behavior that allows it to find an allocation resulting in a 4.85% higher total VoI in longer runs.

Then, the second experiment is to evaluate the performance of GA and QPSO in optimizing the distance based heuristic described in Alg. 1. Given the lower computational complexity of the distance based heuristic with respect to the simplified user-specific VoI model, whose evaluation requires a full simulation run of the testbed scenario within Phileas, the number of iterations for GA and QPSO is set

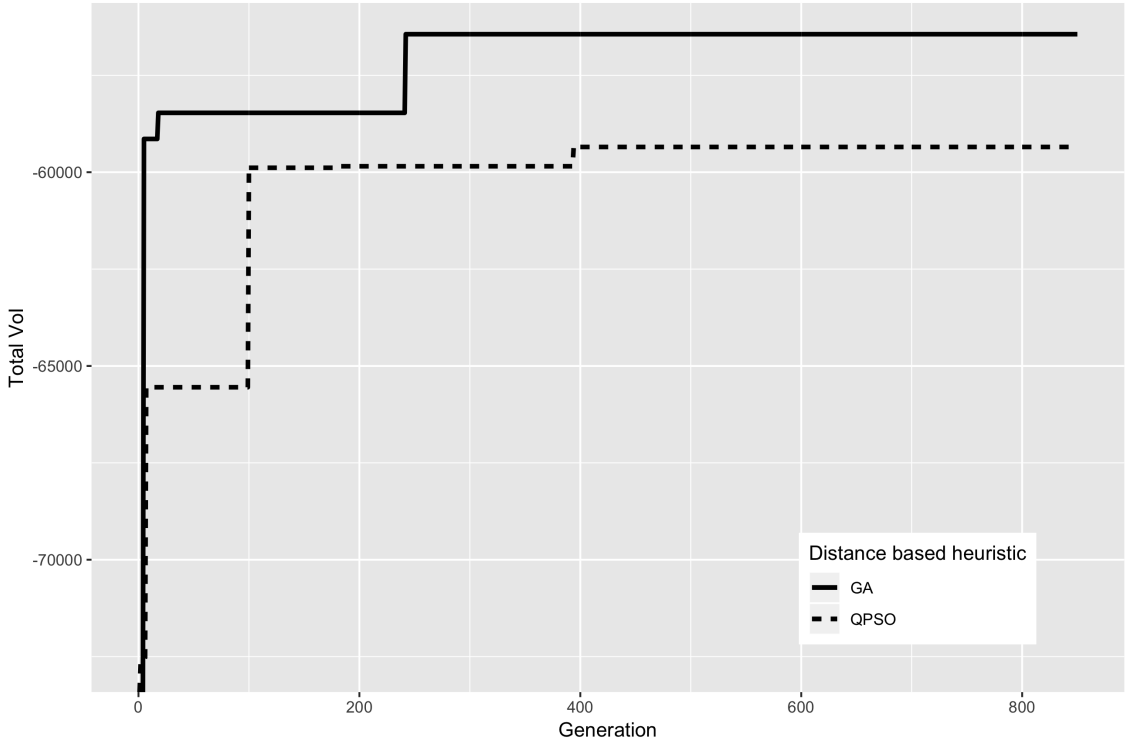


Figure 5.3: Best values for the distance based heuristic using the two meta-heuristics as optimization techniques.

to 850. The same configuration of the previous experiment are used as input to the distance based heuristic.

Fig. 5.3 illustrates the performance of QPSO and GA in optimizing the distance based heuristic. GA outperforms QPSO both from both the exploration and exploitation points of view. More specifically, GA finds an optimal value around the 400-th iteration, while QPSO reaches its maximum around the 800-th iteration. However, both algorithms demonstrate good performance in finding an optimal solution with a tiny gap during the entire process and an average 5% gap at the end of the process.

It is worth noting that, unlike the simplified user-specific VoI model, the execution of the distance based heuristic does not return a VoI value. To make the two approaches comparable, there is the need to evaluate the service component allocations produced by the optimization of the distance based heuristic, use them as input for Phileas simulation runs, and calculating the generated total VoI using the simplified user-specific VoI model of eq. (5.8). The results are depicted in Fig. 5.4, which compares the total VoI values achieved through the optimization of both models. In addition, from the reported results it is possible to verify how the distance based heuristic represents a good approximation of simplified user-specific VoI model.

As illustrated in Fig. 5.4 both GA and PSO find relatively good solutions.

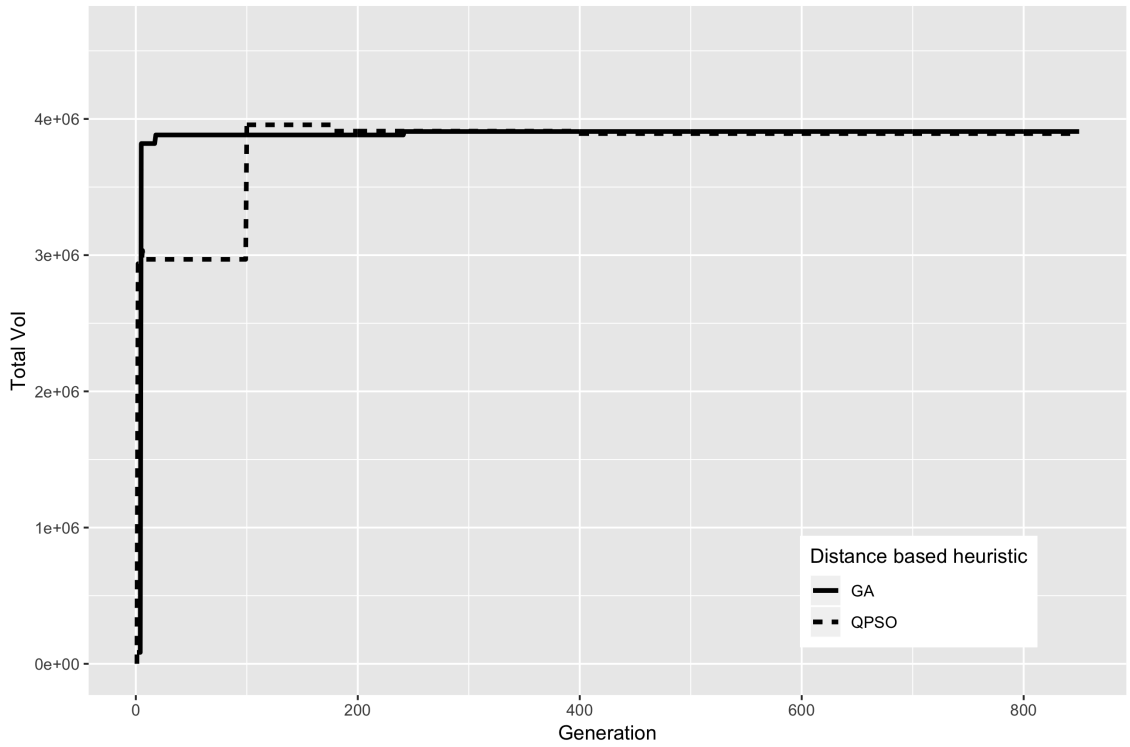
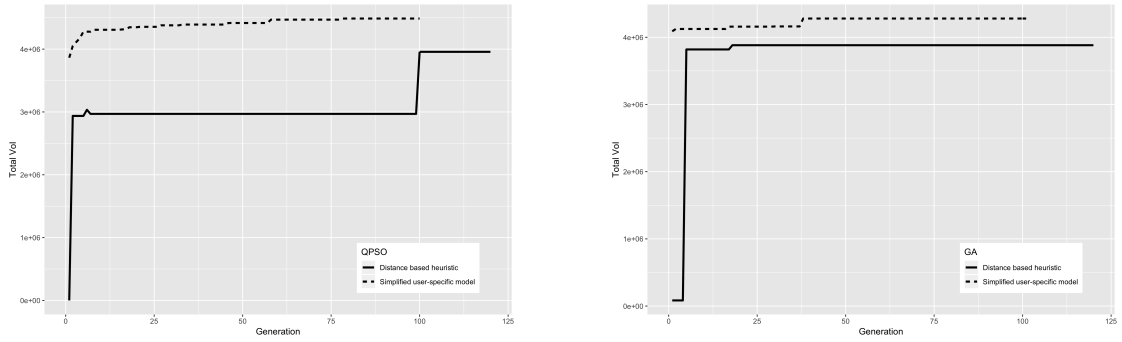


Figure 5.4: Total VoI resulting from the service components allocations generated by the optimization of the distance based heuristic.

GA finds an allocation capable of generating a greater VoI around the first 100-th iterations, while QPSO converges to this result afterward with a 1% average gap the end. It is worth noting that the total VoI value is fluctuating because of the discrepancies between the two models. These results indicate that the distance based heuristic is capable of finding good solutions in term of total VoI generated and demonstrate the soundness of its formulation and the validity of the selected approach. Considering that approximating the VoI function is a challenging task, these results seem to configure the distance based heuristics as a promising approach for all those scenarios requiring a quick allocation or re-allocation of resources.

To better illustrate the gap between the distance based heuristics and the simplified user-specific VoI model, this Section also presents the direct comparison of the previously discussed results in Fig. 5.5a and Fig. 5.5b, considering 125 iterations. At the 100-th iteration, the distance based heuristics reaches about 88% performance for QPSO and 91% for GA when compared to the simplified user-specific VoI model. Therefore, both models are capable of finding good solutions after few iterations of the meta-heuristic algorithm, which is a quite positive result for the distance based heuristics, which is much faster to calculate.

To illustrate the differences in computational complexity between the distance based heuristics and the simplified user-specific VoI model, Table 5.3 reports the configurations and the execution time, measured via the time UNIX utility, to run



(a) QPSO (b) GA

Figure 5.5: Comparison between the simplified user-specific VoI model (QPSO, GA) and the distance based heuristics (A-QPSO, A-GA) using both metaheuristics.

the optimization on both the simplified user-specific VoI model and the distance based heuristic. As it can be seen, the distance based heuristics is much faster than the simplified user-specific VoI model.

Considering the good results provided by the distance based heuristics, it is worth looking for a hybrid approach that can combine this model with 5.8 . The combined model alternatively executes a higher number (850 in the current configuration) of runs of the distance based heuristics and lower number of runs (15 in the current configuration) of the simplified user-specific VoI model. The idea is to use the optimal solution provided by the distance based heuristics as a new starting point for the simplified user-specific VoI model that can later explore a broader search space and avoid local maxima, thus allowing to find improved solutions at lower computational costs. More specifically, the meta-heuristic optimization algorithm is configured to use the best allocation generated at the previous step as a starting point for another 15 iterations of the simplified user-specific VoI based model.

Fig. 5.6 depicts the results achieved by the optimization of the combined models for both QPSO and GA, reported as C-QPSO and C-GA in the legend. To give a graphical representation of the improvements, Fig. 5.6 shows with a red dashed line the best allocation that QPSO found in optimizing the distance based heuristic. More specifically, the combined optimization can improve the results achieved at the previous step of about 8.59% with regards to QPSO and 6.29% for GA. Even in this

Table 5.3: Execution time of the optimization models

Model	M-H	Population Size	Iterations	Execution Time
Distance based heuristics	GA	128	850	6.53s
Distance based heuristics	QPSO	40	850	2.87s
Simplified user-specific	GA	128	100	10030.43s
Simplified user-specific	QPSO	40	100	4113.16s

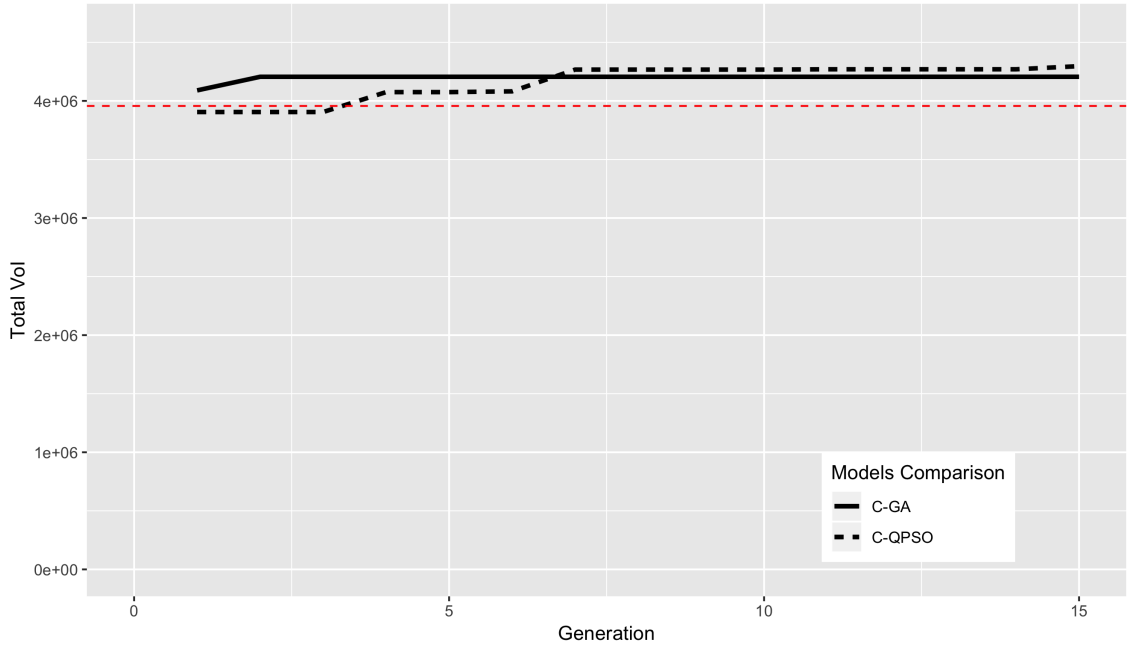


Figure 5.6: Comparison of the combined with the distance based heuristic (red dashed line).

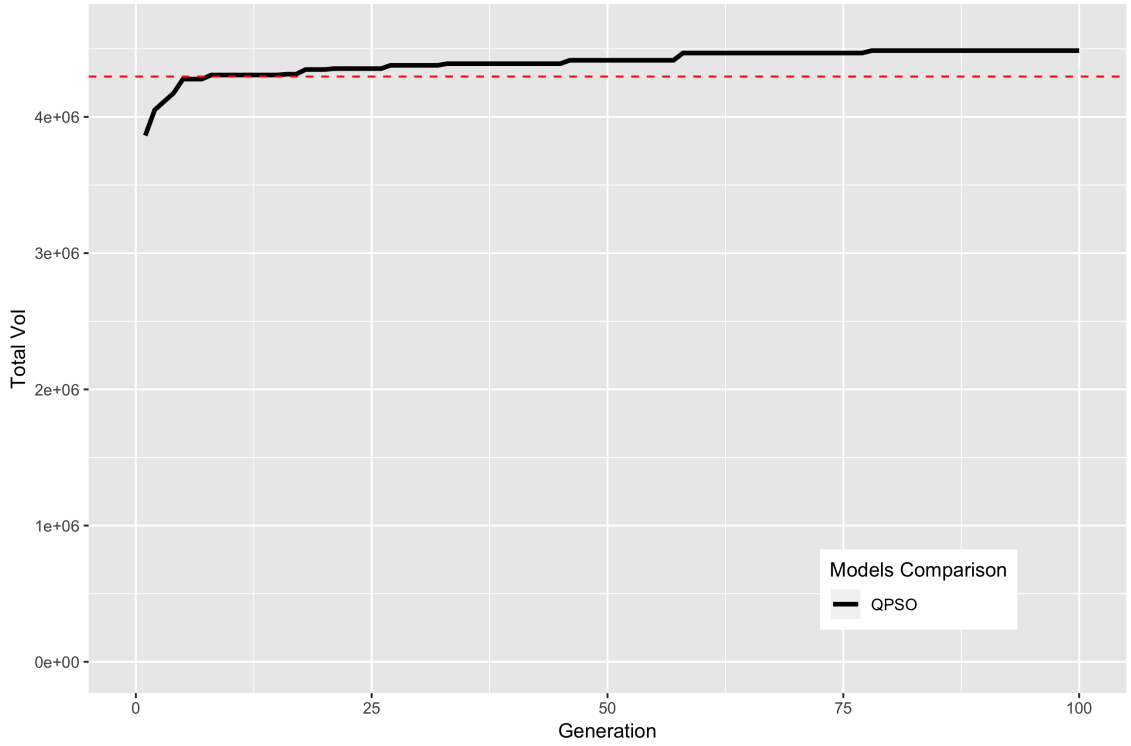


Figure 5.7: Simplified user-specific VoI model versus combined model.

case, QPSO demonstrates faster computational time and better results in general.

The combined model represents a very good trade-off between model complexity and accuracy, which can make the combined model well suited for Fog service fabric management in dynamic scenarios. However, it is worth noting that this responsiveness comes at the price of lower explorative capabilities. To illustrate this, Fig. 5.7 compares the simplified user-specific VoI model with a baseline representing the best service component allocation obtained from the combined model (red dashed lined). The results show that the simplified user-specific VoI model outperforms the combined model of an average 4.43% at the end of the optimization process. This means that less dynamic scenarios in which, for instance, service component allocation could be planned in advance, might be better served by the simplified user-specific VoI model than by the combined model.

Finally, Table 5.4 presents a final report of the best results, in term of VoI collected in the given time window, by the distance based heuristics, simplified user-specific VoI model, and combined model. As discussed, the combined model represents a significant improvement with respect to the distance based heuristics, with a limited increase in computational costs. At the same time, the simplified user-specific VoI model has a higher computational complexity but is capable of identifying the most convenient service component allocations

Table 5.4: Maximum Total VoI for each optimization configuration

	Meta-heuristics	
Model	GA	QPSO
Distance based heuristics	3907701	3956772
Simplified user-specific	4279243	4486806
Combined	4205837	4296572

Chapter 6

Reinforcement Learning for VoI-based optimization

Resource management in Fog and Edge Computing can particularly benefit from the adoption of self-* approaches capable of learning the best resource allocation strategies to adapt to the ever-changing conditions of these environments [65, 66]. In this context, Reinforcement Learning (RL) has been widely investigated from academia as a powerful tool for problem-solving. RL is a goal-oriented training of a software agent that needs to learn the sequence of actions, which maximizes a reward.

Fog and Edge Computing resource management solutions can leverage RL as a powerful tool to deal with the management of dynamic environments requiring continuous adaptions. Interested by the promising capabilities of RL, this Chapter investigates Reinforcement Learning (RL) as an enabling method for online optimization of Fog services [67].

6.1 Need for Continuous Optimization

Resource management solutions for Fog and Edge applications require frequent, and possibly continuous, re-configurations of network and services in order to guarantee high levels of Quality-of-Experience (QoE) and Quality-of-Service (QoS) [68]. In turn, this requires the adoption of relatively complicated continuous optimization solutions, whose proper setup often requires the tuning of several parameters, that further increase the problem complexity.

The solutions described in the previous Chapters build the ground to develop VoI management middlewares capable to optimize the allocation of Fog and Edge processing tasks according to the VoI criterion. However, the illustrated management framework is designed to optimize a quasi static scenario, but real-world applica-

tions will likely operate in very dynamic scenarios. To this end, the distance based heuristic illustrated in the previous Chapter demonstrated to be a good trade-off in finding high VoI service component configurations in a relatively short time, thus making it a good candidate for online optimization.

A different and potentially very promising approach lies in the adoption of self-* approaches, leveraging solutions that are capable of autonomously learning the best strategies to adapt to the current conditions. To this end, a recent trend is the adoption of Reinforcement Learning (RL): an evolving area of machine learning originally inspired by the psychology of animal learning [69]. RL consists of a goal-oriented training of an agent to learn the optimal actions (a policy) to interact with a specific environment. The applicability of RL tools has been investigated in several fields, such as the optimal placement of Network Function Virtualization (NFV) [70, 71], energy-efficient resource allocation [72], and latency minimization [73].

RL can be a valuable building block for Fog and Edge Computing management solutions. Differently from supervised learning methods, which require human intervention for labeling data, RL allows to naturally train a software agent to learn an optimal policy by interacting directly with the environment. This provides a valuable tool to tame the dynamicity of Fog Computing environments, which need continuous interventions to manage the available resources and meet the current applications' requirements [74, 75]. Therefore, Fog Computing management solutions can leverage RL for automatically tuning the configuration parameters when the environment conditions change to guarantee the delivery of the expected QoS and QoE.

Motivated by the promising capabilities of such techniques, this Chapter investigates the application of RL as a candidate for the continuous optimization of a value-based Fog management framework. Towards that goal, we focus on Value-of-Information (VoI) optimization as resource management criterion for Fog Computing applications.

6.2 RL for Resource Management

RL is an evolving field of machine learning consisting in a goal-oriented training in which an agent interacts with an environment to learn the best possible actions leading to a specific goal [69, 76]. At each action is assigned an immediate reward and the goal of the agent is to learn a policy that would maximize the sum of those rewards. In order to learn a good rewarding policy, the agent needs to solve the same task multiple times in which it will be capable of exploring a consistent number of states.

RL has been applied to different sort of problems from learning how to play classic

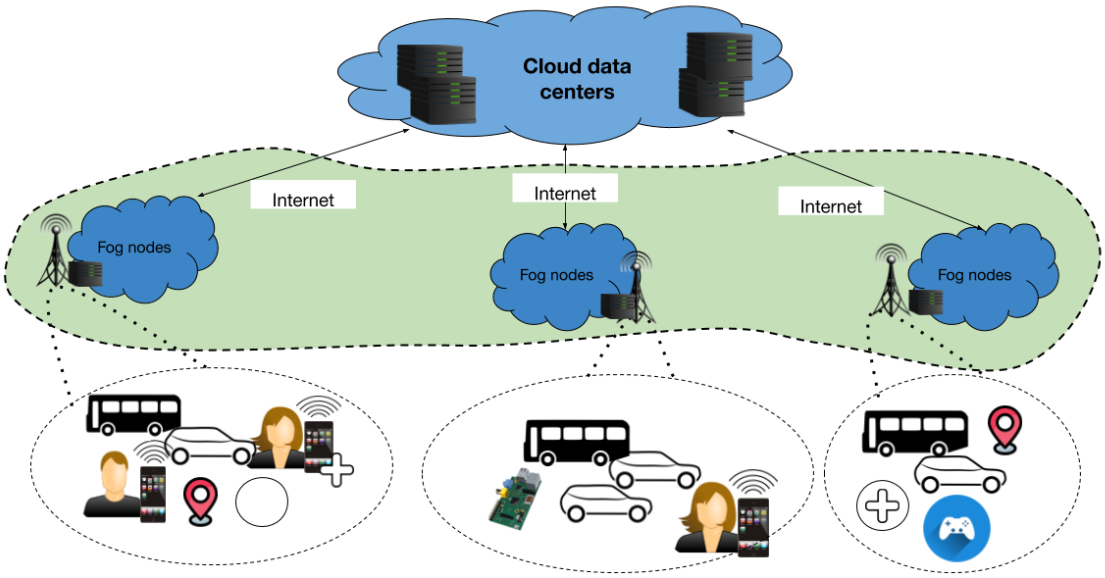


Figure 6.1: A typical Fog Computing environment for Smart City scenarios depicting Cloud data centers, Fog nodes, and IoT end-users.

games to network and service management for dealing with the best allocation of network slices [74, 75], the best placement for Virtual Network Function (VNF) or task offloading in Multi-Access Edge Computing [77, 78] to minimize end-user latency and to maximize the Quality-of-Experience (QoE). Dab et al. formalize an RL problem to learn the best offloading decisions in order to minimize energy consumption on the devices-side under latency constraints for 5G applications in [77]. The proposed strategy is evaluated with extensive simulations in NS3 and proved to be successful in reducing computation time and find near-optimal solutions. Long-Term latency minimization for Fog Computing is also discussed in [73]. In particular, this paper proposes an RL approach combined with evolution strategies to deal with real-time task assignment that allows reducing computation latency in the long-term period.

In [78], Nakanoya et al. propose an interesting technique for applying RL to online optimization of Virtualized Network Functions (VNF) sizing and placement. In particular, the authors propose a two-step RL that divides the learning process into two phases with the aim of reducing the learning exploration steps. The proposed method is evaluated under three different realistic network environments and proved to be cost-effective. In [74] Chen et al. discuss DRL for task offloading decision in MEC for mobile ultra-dense in sliced RAN. More specifically, this paper proposes two Double DQN learning algorithms for solving computational offloading under dynamic network conditions. The authors in [79] discuss a Q-learning based load-balancing algorithm for Fog Networks that allows reducing the processing time and overload probability.

Li et al. investigate RL for Network Slicing in [80] by illustrating general concepts and applications for resource management in network slicing by comparing different scheduling algorithms. An interesting formulation leveraging a Deep-Q Network (DQN) is the one discussed in [81]. In this work, the authors propose an algorithm called JTOBA that iteratively solves the problem of best joint task offloading and bandwidth allocation in MEC. A very interesting approach is to change the initialization state every 100 episodes with the current best result. A different RL application is described in [82], in which a fast task allocation (FTA) algorithm leveraging DRL is proposed to allocate tasks among heterogeneous UAVs. The authors claim that FTA is highly adaptive even in case of different set of tasks and environment variability.

6.3 RL as another Optimization Tool

The previous Chapter enquired that the practical adoption of a high complexity model requires *simulation-based optimization* [58] to evaluate a particular service components allocation in terms of VoI. Following this direction, the model defined in (5.8) has been implemented within the Phileas simulator [62] and optimized using meta-heuristics techniques. Instead, this chapter presents a different methodology to optimize the value of (5.8) using RL techniques.

6.3.1 The MDP for VoI allocation

In order to exploit RL techniques into a Fog Service Management Framework, it is worth defining a Markov Decision Process (MDP) for the system model [83]. An MDP is a general framework (particularly suited for RL) for defining decision making problems [69]. With this regard, a proper modeling of states and actions is essential for RL tasks. In fact, RL in general requires a good knowledge of the whole problem and a proper reward definition in order to allow the training process to converge.

The MDP for VoI allocation defines a set $\mathcal{S}$ of states s, a set $\mathcal{A}$ of actions that allow an agent to move from a state s to another state s', and a Reward policy R the defines the reward given by an action that moves the environment into a different state. In particular, $R_a(s, s')$ is the immediate reward for performing action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$. Finally, the goal of the MDP is to find the optimal policy $a = \pi(s)$ that gives the best action $a \in \mathcal{A}$ under state $\in \mathcal{S}$ that allows to maximize the Q-function $Q(s, a)$ for each state action pair.

With regards to a state $s \in \mathcal{S}$, we define it as an array-like service component allocation $s = \{sc_1, sc_2, ..., sc_n\}$ in which the value of the i-th element $i - element$

represents the fog device d_j where the service component sc_i is allocated. For instance, {3, 2, 4 .., k, 3 } is an example of a state $s \in \mathcal{S}$, where the service component sc_1 (the first element of the state array) is allocated on fog device d_3 and k is $|\mathbb{D}|$, the number of fog devices.

On the other hand, we define an action $a \in \mathcal{A}$ as the selection of a fog device $d \in \mathbb{D}$ where to allocate a service component sc_i on. For example, an action $a = 2$ indicates to allocate the service component on fog device d_2. Furthermore, let us note that, because the state and action spaces are finite, the formalized MDP is finite MDP.

More specifically, the VoI allocation MDP consists of a sequence of n discrete time steps $t = 0, 1, 2, \cdot, i, \cdot, n$ in which an agent analyzes each service component sc_i (where the i index corresponds to the i time step) and decides whatever or not allocates it on a different fog device d_j. When an action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$ is performed, a new state s' is reached, and the agent gets an immediate reward $R_a(s, s')$. Finally, for modeling rewards, this formulation adopts the simplistic but effective strategy of assigning a reward of 1 to those actions that define an allocation capable of improving the value of (5.8) with respect to previous state, and 0 to the others.

6.4 The FogReinForce DRL Solution

Having defined the MDP formulation for the VoI allocation of service components allows us to adopt a suitable RL algorithm for optimizing the problem. Given the dimension of the state space, which grows exponentially with the number of service components to allocate, this Section investigates DRL solutions to propose a Deep Q-Network (DQN) based learning algorithm FogReinForce. Finally, this Section discusses the adoption of FogReinForce within a management framework for enabling the continuous optimization of Fog Computing resource management.

6.4.1 Deep Q-Network Algorithms

Classical RL Q-learning algorithms requires to explore over the state space to learn the optimal Q-value $Q(s, a)$ for each possible state s and action a by storing all transitions in Q table. However, when the state space's size is large, learning a Q-value in a tabular encoding function is not tractable for both memory and time limits.

To overcome this limitation, several DRL algorithms have been proposed in literature. A widely used Q-learning algorithms is DQN, in which the prefix Deep is to introduce the adoption of neural-networks for leaning a parameterized estimation

the Q-value function $Q(s, a, w)$. In more words, the DQN algorithm substitutes the Q-table with a Deep Neural Network for mapping states into action values.

For classical Q-learning algorithm, the learning process updates the Q-value function in a iterative way using the Bellman equation as follows [69]:

$$Q(s,a) = Q(s,a) + lr[R(s,a) + \gamma \max(Q\prime(s\prime, a\prime) - Q(s,a)]. \tag{6.1}$$

where lr is the learning rate, $R(s,a)$ the immediate reward for performing action a under state s, γ the discount rate for taking into account the expected future reward given by state $s\prime$ and all consequent possible actions.

Instead, for DQN the goal is to minimize a loss function defined as the squared difference from the target and predicted value:

$$Loss = (R(s,a) + \gamma \max_{a\prime} Q(s\prime, a\prime; w\prime) - Q(s,a;w))^2. \tag{6.2}$$

where w and $w\prime$ represent weight parameters (for the neural networks), $Q(s\prime, a\prime; w\prime)$ indicates the target value for the state $s\prime$, and $Q(s,a;w)$ is the current predicted value. During the iterations of the training process the Loss function is minimized using the gradient descent method with respect to the parameters w.

Given the definition of the Loss function, a DQN algorithm update the weights according to the formula:

$$\begin{aligned} &w_{t+1} = w_t + \\ &lr[R(s,a) + \gamma[\max(Q\prime(s\prime, a\prime; w) - Q(s,a;w)]\nabla_w Q(s,a;w) \end{aligned}. \tag{6.3}$$

To learn the parameterized estimation of the Q-value function, DQN exploits two different Q-value networks: a local Q-network and a target Q-network. During the training process, the target network is updated after a configurable amount of iterations with the weights from the local Q-network to stabilize the learning process. Finally, most recent DQN versions rely on a replay memory to store some state transitions for training. The algorithm randomly samples a mini-batch transitions from the replay memory to update the neural network, thus reducing the correlation and enabling experience replay.

6.4.2 FogReinForce Learning Algorithm

To solve the VoI allocation MDP problem, we propose the FogReinForce learning algorithm, which exploits DQN for implementing the learning process. FogReinForce aims at finding the VoI optimal allocation for service components by learning the optimal policy $a = \pi(s)$ that selects the best actions to take under a particular state

$s \in \mathcal{S}$ to obtain a performing service components allocation in a finite number of steps N.

Algorithm 2: FogReinForce pseudo-algorithm

```
Data: ϵ, γ,update_step
Result: π a policy for mapping states into actions
Q_0(s, a) = 0 initialize action-value function to 0;
initialize target action-value Q-network for Q estimation;
initialize replay memory RM;
initialize a random allocation s_r to be used as initialization state;
for i in 1, max_episodes do
    initialize state s_1 = s_r;
    initialize er = 0 episode reward ;
    for t in 1, N do
        if rand() ≥ ϵ then
            select action a_t = argmax{Q(s_t, a)};
        else
            select a random action a_t;
        take action a_t and generate the next state s_{t+1};
        evaluate s_{t+1} using (5.8) and calculate the reward r_t;
        set s_t as the next state s_{t+1};
        store transition in replay memory RM;
        update er;
        if t is an update step and there enough transitions in memory then
            sample a random subset of transitions from RM and learn;
        else
            calculate discounted reward using γ;
    decrease ϵ and make it decay;
```

Alg. 2 gives a simplified illustration of the FogReinForce algorithm, which mainly recalls a classical DQN algorithm with experience replay. Alg. 2 takes as input the values for ϵ, γ, and the update_step used during the training. The replay memory RM is used to store the transitions and for sampling a mini-batch during the training with experience replay [84].

As depicted in Alg. 2, FogReinForce defines the allocation problem as an episodic task with a maximum number of episodes $max\_episode$. More specifically, FogReinForce defines an episode as a finite number of steps N, after which the episode is considered finished. The number of steps is set to $n = 2 \times k$, where $k = |SC|$ is the number of service components to allocate.

At the beginning of a new episode, the initial state s_1 is reset to the random state s_r generated in the initialization phase of the algorithm. This is a common practice when defining a DRL training process, however, different policies can be chosen. During each step, the agent interacts with the environment by selecting at

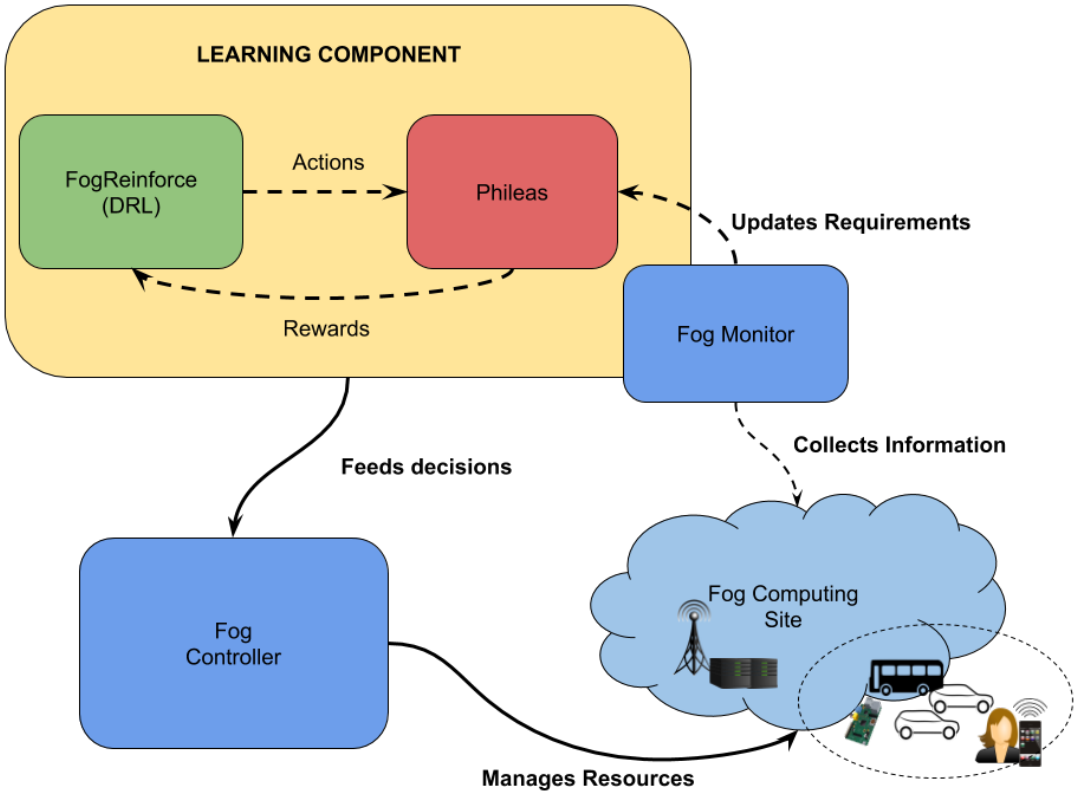


Figure 6.2: An overview of the management framework for enabling the continuous optimization of Fog Computing environments.

step t a fog device for the service component sc_t. The new state is evaluated using (5.8) to calculate the total VoI that new state s_{t+1} generates. During these steps, the goal of the agent is to maximize the cumulative reward over the episode given a finite amount of actions.

6.4.3 Continuous Optimization for Fog Computing

To illustrate readers an architecture for realizing continuous optimization of Fog Computing resource management this Section presents a proof-of-concept framework that incorporates FogReinForce into a learning component that continuously checks for improvements. To this end, Fig. 6.2 depicts the envisioned architecture for allowing continuous optimization of Fog services running in a Fog Computing site. More specifically, the Fog management framework includes three main components: a learning component, a Fog controller, and a Fog monitor.

Firstly, the Fog monitor is responsible for collecting information regarding a Fog Computing site including users' interests, available fog devices, running services, and network conditions. In addition, the Fog monitor updates the collected information into the Learning Component to create an input configuration for the Phileas

simulator.

The Learning component is to find the optimal VoI allocation of service components on the resources available at the Fog site. To this end, the Phileas simulator is configured with the data collected by the Fog monitor to be a realistic representation of the environment at the Fog Computing site. In order to achieve the optimal allocation of service components on Fog devices, the Learning Component runs FogReinForce for a finite number of episodes. In this way, FogReinForce interacts with the Phileas simulator to find the service components configuration that maximizes the feedback representing the total VoI delivered during the simulation.

When the Learning component finds a suitable allocation, it instructs the Fog Controller where to allocate the service components running at the Fog Site. Finally, for enabling online optimization, this loop runs continuously, thus ensuring to tackle the dynamicity of Fog Computing environments and deal with users' mobility, system's load, and other conditions.

6.5 Evaluation

The following evaluation relies on the same Fog Computing scenario described in the previous Chapter. This would provide readers a comparison with the results illustrated in the previous Chapter. The scenario is a fictional representation of a Smart City use-case, in which smart citizens exploit the functionality of different Fog services provided by the municipality. More specifically, the testbed contains the description of 7 devices, 9 data sources, 4 user groups, and 8 service components.

To reenact realistic conditions, the locations of devices, data sources, and user groups are defined in the Smart City scenario with a latitude and longitude position in accordance with the GPS position system. This allows setting fixed communication ranges for users and devices for reenacting wireless communications. Furthermore, this would drive FogReinForce to allocate service components according to a minimum distance policy, e.g. privileging those allocations that process data messages close to where they will be consumed.

The aim of these experiments is to verify the capabilities of FogReinForce in optimizing the total VoI delivered to the end-users of Fog services during a fixed time-window. More specifically, FogReinForce will learn how to configure service components on the available fog devices using the total VoI delivered as feedback.

6.5.1 Reinforcement Learning Configuration

FogReinForce leverages on a Deep Q-Learning Network (DQN) implemented in Python, which exploits the pytorch library[1] for implementing the training of the Deep Q-network (neural-network) responsible for mapping states in action values. To this end, the Python programming language it is a valuable tool for experiencing with machine learning and data analysis. In addition, the pytorch library provides a user-friedly API for implementing the state-of-the-art machine learning models and optimization algorithms.

The Q-Network is implemented as a two-layers neural network with 64 nodes for each hidden layer, a discount ratee $\gamma = 0.95$ to determine the present value of future rewards, a learning rate of $lr = 0.0005$, and ϵ value starting from 1 and annealing to 0.01. Finally, the mini-batch size for experience replay is set to 64.

During the training phase, FogReinForce will interact with the Phileas simulator written in Ruby in a HTTP fashion. More specifically, given a state s representing a service components allocation, FogReinForce requests (via HTTP) to Phileas to calculate the value of (5.8) for the given configuration. Even if performance-wise is not great, this is a common practice to integrate software components written in different programming languages. Future versions of this optimization framework will provide a better integration of the DRL capabilities within the VoI management framework to reduce the required training time and speed-up the entire learning process.

With regards to the training phase configuration, the number of episodes is set to 1000 to give the agent a fair amount of iterations to learn an optimal policy $a = \pi(s)$. In addition, each episode is defined as a sequence of 16 steps after which the episode is considered done. As described in the previous Section, if the action of allocating a service component to another fog device (different configuration) brings to a more performing allocation (in terms of total generated VoI) the agent gets a reward of 1, otherwise it gets 0.

As first state FogReinForce selects a random allocation of service components on fog devices. Other options such as a greedy to calculate a feasible starting state are possible. However, opting for a random state is a good assumption to verify if given a fair amount of training iterations, the agent would learning a sequence of actions leading to a good rewarding state. In fact, the agent should be capable of learning from whatever starting point given a finite number of steps and episodes [69].

[1] `https://pytorch.org`

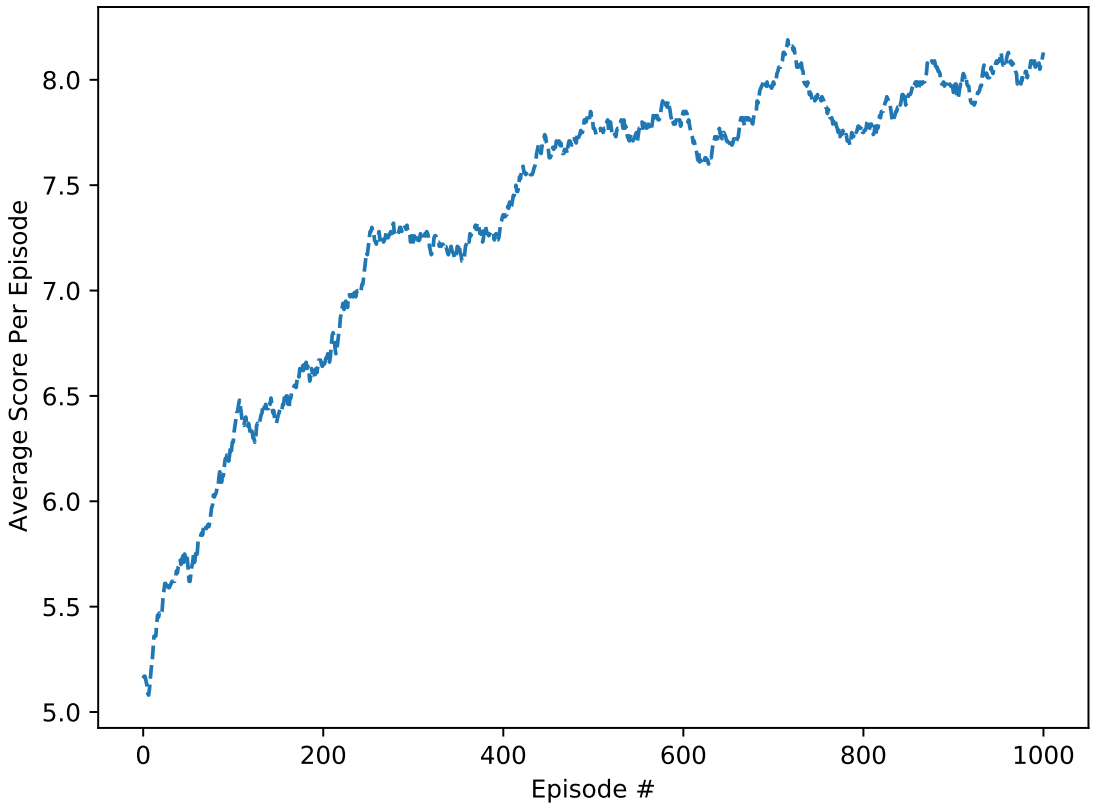


Figure 6.3: The training of the DRL agent over the VoI model for 1000 episodes.

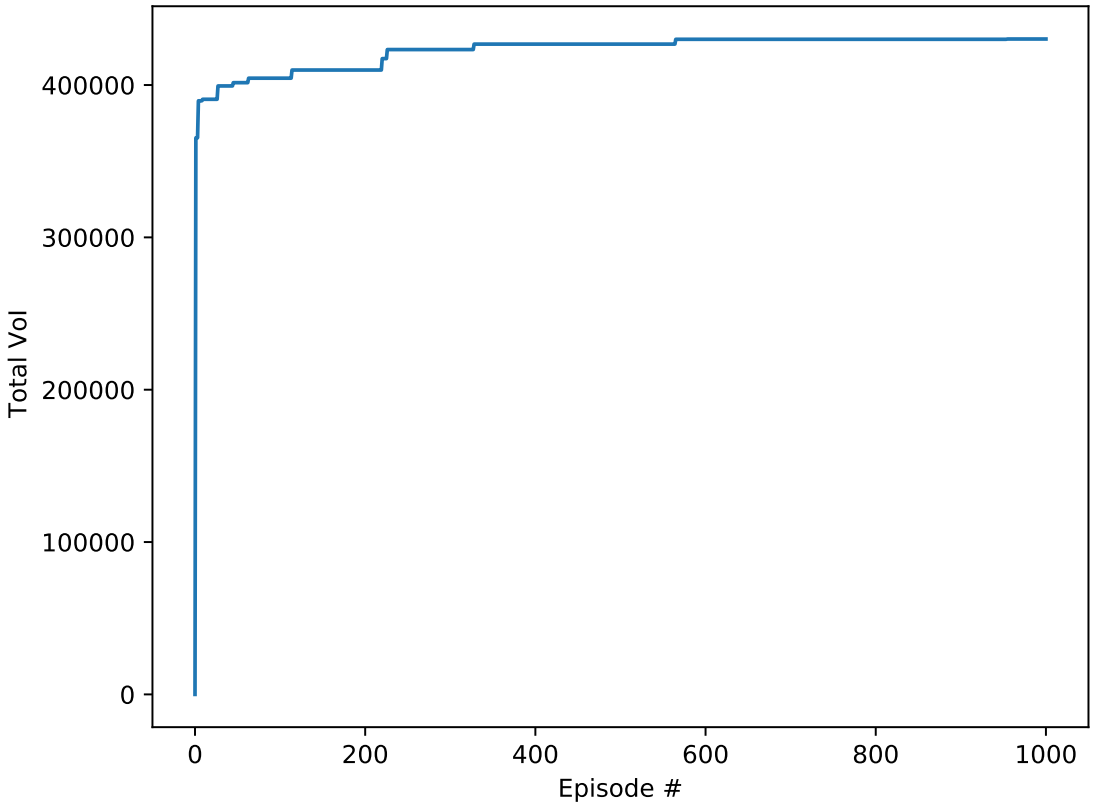


Figure 6.4: The best values for the VoI model during the training phase.

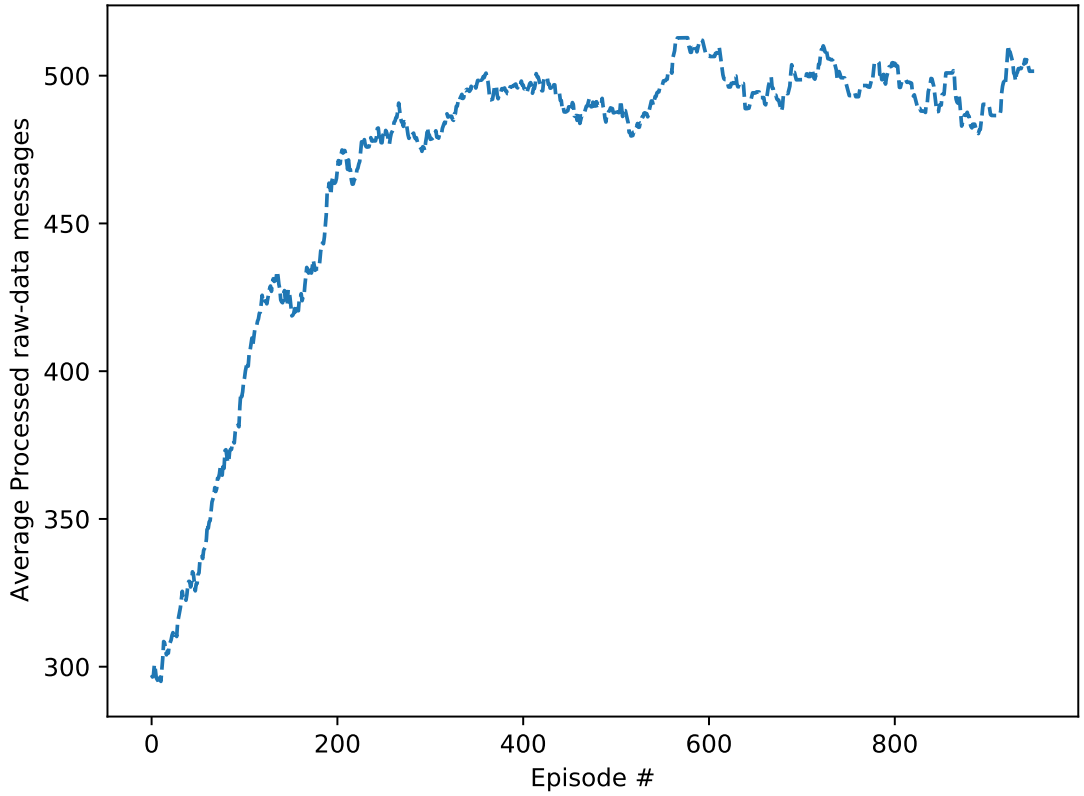


Figure 6.5: The number of processed raw-data messages during the training phase.

6.5.2 Results

FogReinForce is configured with the parameters discussed in Section 6.5.1 to learn the optimal policy $a = \pi(s)$ that would maximize the total VoI (5.8) delivered within a specific time window. To this purpose, Phileas simulator reenacts the fictional Smart City scenario for a limited time window of six minutes. This is to reduce the training time and to allow the evaluation of different service component configurations. As illustrated in Alg. 2, at the beginning of each episode, FogReinForce initializes the initial state using a randomly generated state. Then, at each step, it learns the actions that maximize the cumulative reward of the episode.

During the training phase, for each episode, the score, i.e. the sum of rewards, and the state $s_{optimal}$ which leads to the best value of (5.8) have been collected.

Fig. 6.3 reports the training phase of FogReinForce to optimize the VoI model defined in (5.8). More specifically, Fig. 6.3 depicts the time-series of the average scores that FogReinForce achieved during the 1000 episodes of the learning process. The reported trend is increasing, thus indicating that the algorithm is capable of learning a good rewarding policy. However, as common for RL and DRL, the score value has ups and downs during the training in contrast to other machine learning techniques. Given a finite number of 16 steps, FogReinForce can improve the current

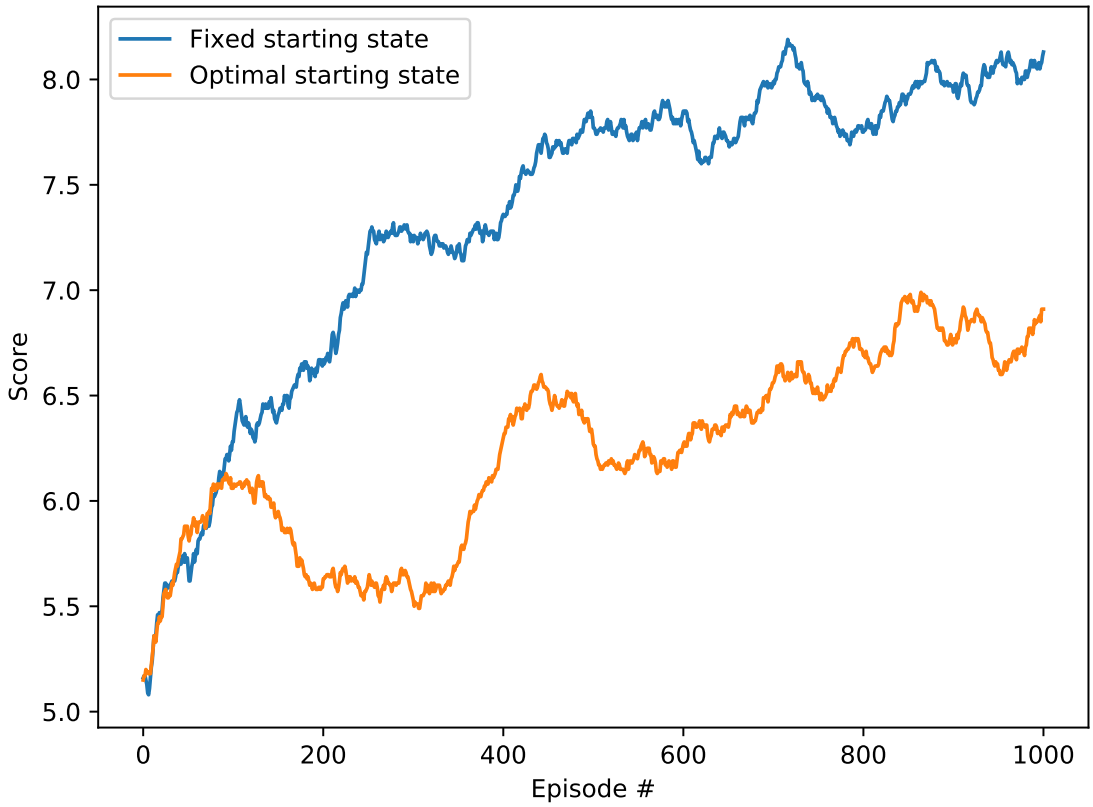


Figure 6.6: The comparison of the two versions of FogReinForce during the training of the VoI model for 1000 episodes.

state up to 12 times for an episode. Moreover, it achieves a cumulative reward value of 8 after only 500 episodes. This is an encouraging result, which demonstrates the viability of DRL methodologies for our particular VoI management framework.

On the other hand, Fig. 6.4 illustrates the best VoI values for (5.8) achieved during the training of FogReinForce. It is worth to note, how the best value is achieved around the 600-th episode, thus indicating that FogReinForce is capable of improving the value of (5.8) in a relatively limited number of iterations. This also shows that FogReinForce can find a good rewarding policy to improve the allocation described by the random starting state s_r.

Another proof that FogReinForce is capable of improving the management of the processing resource at the edge comes from Fig. 6.5. More specifically, Fig. 6.5 depicts the number of raw-data messages that the fog devices can process during the time-window defined by the simulation time. It is worth noting how this number increases during the episodes and that this increasing trend is also related with more optimal VoI allocations. Finally, this demonstrated that VoI methodologies and tools are beneficial for addressing resource management.

As another experiment, it is worth considering a slightly different version of FogReinForce, which updates the starting state s_1 every 100 episodes using the state

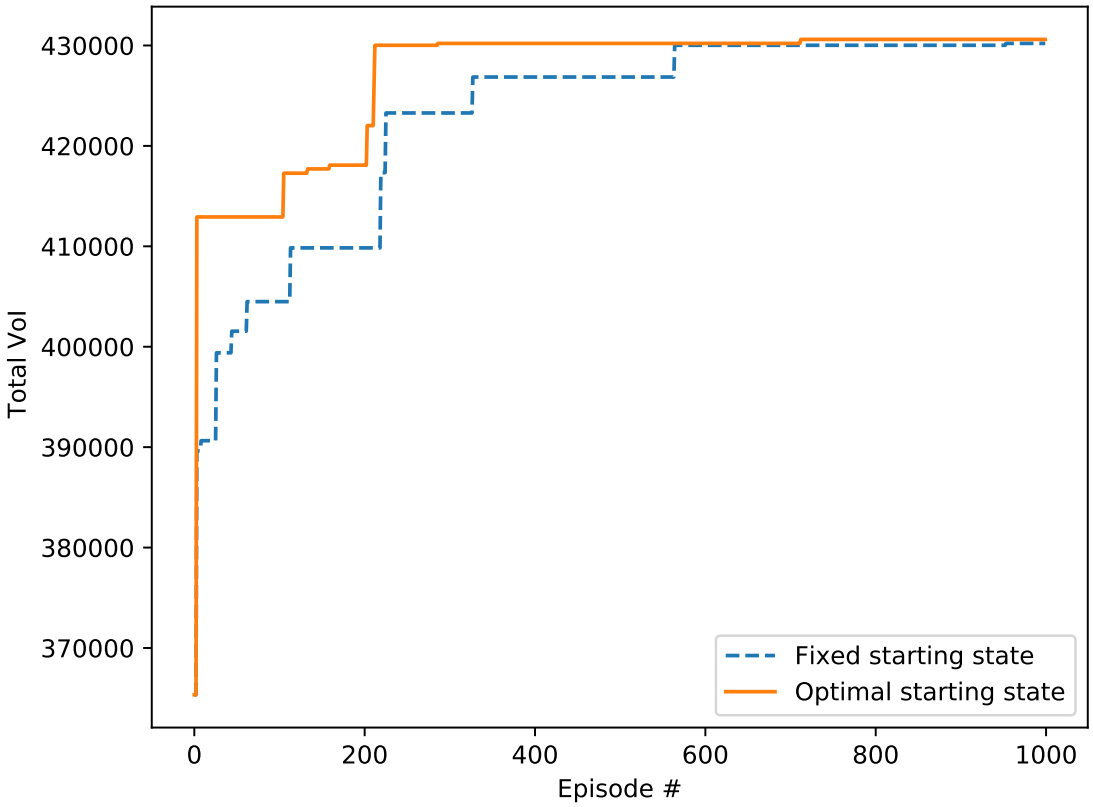


Figure 6.7: The best values for the VoI model during the training phase of the two versions of FogReinForce.

$s_{optimal}$ that generated the highest value of (5.8), inspired by the work in [81]. This is to prove the online optimization capabilities of FogReinForce, which can exploit the experience achieved so far to improve a new service components configuration.

To this end, Fig. 6.6 shows that the traditional version (blue curve) achieves higher scores during training. However, this is trivial because the number of improvements during the training process is a function of the starting state s_1, i.e, by increasing the quality of the starting state the number of possible improvements would decrease. Therefore, this indicates that the number of steps required to improve the current allocation decreases when the starting state is a good service components configuration in terms of (5.8).

On the other hand, Fig. 6.7 highlights the best values for (5.8) achieved during the training of the two versions of FogReinForce. It is worth noting how the modified version converges to the maximum VoI value faster when compared to the traditional version. This is another encouraging result that would allow improving the optimization capabilities of FogReinForce in future works.

Finally, Table 6.1 provides a summary of the best results and compares the two different versions of FogReinForce. Both versions achieve a quasi-equal maximum for (5.8). It is worth noting that the slight difference is due to some randomness

Table 6.1: Summary of Results.

	Fixed starting state	**Optimal starting state**
Max VoI	430207.47	430680.14
Max Improving steps	12	10

implemented in the simulator despite the efforts to initialize all random variables with a fixed seed. On the other hand, the amount of improving steps during the training is equal to 12 for the fixed starting state version and 10 for the other one.

Chapter 7

An Holistic view for Fog Computing Management

Global management of both applications and networks resources can help to provide enhanced QoS and QoE. In this area of research, Software-Defined-Networking has been widely explored during the last decade as promising technology for network management [85, 86]. SDN provides a logical separation of control and data plane to decouple the forwarding of the network packets from their routing. Therefore, SDN is a key enabling technology for the management of complex networks. Given its interesting applications, researchers have also investigated its adoption for managing Fog Computing networks. However, there is still wide room for investigation, especially when it comes to integrate both the applications and networks requirements into the management of Fog Computing.

With the goal of filling this gap, this Chapter describes the experiences in developing the Holistic Processing and Network (HORNET) solution for allowing an optimal management of both devices and networks in Fog Computing. HORNET is a proof-of-concept middleware that provides an overall set of functionality to deal with the requirements of Fog Computing environments [87]. HORNET adopts the AIV model for managing information processing/dissemination in a dynamic and integrated way. In addition, to deal with the high heterogeneity of Fog Computing environments, HORNET supports a Multi Layer Routing (MLR) approach to exploit multiple routing options at different abstraction levels at the same time.

To better illustrate the HORNET solution, this Chapter first introduces the concepts of SDN and NFV and their applications to resource management. Then, it discusses the motivations and the role of a novel solution capable of considering both application and network requirements for Fog Computing management. Finally, this Chapter provides extensive experimental evaluation both on a real testbed and in a simulated environment.

7.1 SDN and NFV for Management

Among the efforts into Fog Computing related research, SDN technologies is a compelling topic. In fact, the adoption of SDN among with network virtualization tools would allow to manage a coordinated orchestration of service functions to provide augmented QoS and QoE. In this field of research, SDN can leave rage the VoI concept as prioritization criterion for traffic management. This Section discusses the related work for this field before presenting the HORNET solution.

While the SDN approach has been traditionally adopted (and by now it is considered the standard solution) in datacenters and huge enterprise networks [88], some first research efforts demonstrate its validity also in Fog environments [26, 89, 90, 91, 92]. To this purpose, the section revises state-of-the-art contributions along three primary directions: i) the current literature proposing to adopt the SDN approach in Fog environments, ii) the recent trend towards softwarization of network services taking advantage of the SDN, and iii) dynamic service composition and task offloading to better manage Fog nodes and related computational/resources, eventually also adopting Quality-of-Information (QoI) and Value-of-Information (VoI) concepts.

As presented in a recent survey [93], most of SDN-based Fog routing solutions propose the adoption of SDN technologies in Fog Computing to provide efficient routing mechanisms capable of addressing low latency, low bandwidth, and security requirements of Fog Computing environments. In addition, the paper presents an SDN solution based on a hierarchy of Fog controllers distributed between the Cloud and the edge; the primary idea is that frequent events are locally managed on Fog controllers and rare events are globally managed by the Cloud controller. Other papers focus on security aspects related to the adoption of SDN in Fog Computing. For instance, [94] analyzes security vulnerabilities of the OpenFlow channels and proposes an attack model and related countermeasure based on Bloom filters. Moreover, [92] proposes a novel approach based on the combination of SDN and Blockchain to securely manage computational resources (Fog nodes) at the edge, with the primary goal of exploiting the Blockchain to connect together multiple SDN controllers deployed in a distributed manner in the Fog layer. Finally, [95] outlines how another relevant application of SDN in Fog Computing can be the identification of threats and anomalies. In particular, the paper exploits the SDN control plane to dynamically deploy software agents monitoring the traffic to identify anomalies and eventual attacks.

The wide adoption of SDN together with Network Function Virtualization (NFV) has more recently pushed the attention on the softwarization of network services traditionally implemented in hardware [96, 97]. In this scope, Service Function

Chaining (SFC) aims at providing an efficient composition and/or orchestration of different and related network applications/functions to achieve a chain of services (building block services) suitable for devices and users in Fog Computing environments. In [98], Yu et al. propose the adoption of SDN as the enabling methodology for QoS traffic steering in SFC. More specifically, the authors give a mathematical definition and formulate a polynomial time approximation algorithm. Instead, Zhang et al. discuss rule management in SDN-based IoT and propose to aggregate and minimize the number of forwarding rules by multiplexing different traffic flows flowing in the same path aggregating them with a VLAN ID label [99]. In [100], the authors describe a network function virtualization-aware orchestration for SFC placement in the Cloud. In particular, the authors propose an heuristic for component orchestration in small- and large-scale DC network (BACON) that minimizes the intra- and end-to-end latency of the SFC. A similar approach is given in [101], where authors formulate an optimization problem for the deployment of service function chain in 5G mobile networks. Furthermore, [102] presents an interesting survey focusing on Fog Computing applications, pointing out some of the current challenges that orchestration techniques usually find in Fog Computing scenarios: churn and unreliability of nodes at the edge, heterogeneity of resources, security and privacy related issues, and location issues.

Finally, some works focus on dynamic service deployment [103] and data management optimization in [104] in IoT environments. For instance, [105] proposes Data-intensive Service Edge deployment scheme based on Genetic Algorithm (DSEGA) to identify a perfect fit for component services and data deployment on edge nodes in relation to storage constraints and load balancing conditions. Instead, [106] presents a work offload solution considering the geographic location of mobile edge servers and IoT devices to better serve service subscribers, also to decide if and when a task should be run locally or on a remote edge node. Moreover, [107] introduces an innovative approach to predict the QoS in IoT environments based on Neural Collaborative Filtering (NCF) and fuzzy clustering. The main idea is to cluster contextual information and exploit a new combined similarity computation method to identify latent features in historical QoS data.

In conclusion, the SDN adoption on the edge side of Fog environments requires more efforts since there is the need to adopt a wider and holistic point of view. The objective is to support Fog service reconfiguration not only considering networking features and resources (such as most of the current literature propose) but also service composition based on the aggregation of processed data (as described in this chapter).

7.2 Joint Management of Transport and Application Layers

Fog Computing applications would significantly benefit from innovative solutions specifically designed to consider the time-sensitive, location-aware, and information-centric nature of IT services and the scarcity of resources. In this context, approaches that jointly address the issues of information prioritization for processing and dissemination and of traffic engineering seem particularly well suited to address the intrinsic dynamicity of Fog Computing scenarios.

To explore the need of joint management of both information processing and dissemination, this Chapter proposes the *HOlistic pRocessing and NETworking (HORNET)* solution [87]. HORNET adopts an innovative holistic SDN-based approach that simultaneously and jointly considers both transport (for packet re-routing and engineering) and application (for dynamic deployment and de/activation of services) layers. This represents a significant difference from traditional SDN approaches focusing only on the networking perspective and allows HORNET to perform an optimal management of both computational resources and network flows configuration.

Within this general approach, HORNET adopts the AIV service model which was specifically designed to address the requirements and challenges of Fog computing applications. First, the HORNET service model follows an "acceptable lossiness" perspective to Fog service development, at both the processing and communication levels. The assumption is that Fog environments are resource scarce and might not be able to fully support application requirements. For this reason, services should focus their effort on the processing of a subset of data that allows them to maintain acceptable Quality of Experience (QoE) levels, eventually dropping packets related to data deemed as less crucial.

According to the AIV model, service components can be quickly chained to create applications, and just as quickly rearranged in case of need. It is worth noting how this paradigm has similarities with the research on SDN and Network Function Virtualization (NFV), and extends the service chaining concept as a composition of different Virtual Network Functions (VNFs) to a higher-level application-driven perspective, by considering information-centric service components instead of VNFs. In addition, HORNET addresses the dynamicity of the application scenario by implementing at the single service component level a processing logic which is adaptive and capable of reducing requirements to match the level of resources currently available in the deployment location, i.e., the edge device on which the component is running.

To dynamically control and optimize information dissemination between service components, HORNET also introduces an important innovation. More specifically,

HORNET leverages Multi Layer Routing (MLR), which provides three different forwarding dissemination tools to tackle the needs of different types of applications. MLR dynamically uses multiple routing solutions at different abstraction layers, ranging from native IP ones to more expressive but resource-consuming packet dispatching techniques, also considering the actual content within packet payload. The adoption of MLR allows HORNET to tailor the specific dissemination solution for processed information according to service requirements and the current execution context, by dynamically choosing the most suitable among a wide array of different options.

Finally, HORNET adopts VoI as a unifying criterion for the optimization of resource allocation and management. Compared to other criteria, VoI naturally enables a much more effective usage of the scarce and heterogeneous computational and bandwidth resources in information processing and dissemination [21, 108].

The adoption of these innovative models provides to HORNET several advantages compared with traditional solutions. It allows to achieve the desired Quality of Experience (QoE) levels by focusing on the processing and dissemination of the most valuable pieces of information from the end users' point of view, possibly delaying or even dropping less valuable ones.

However, since it is a radical change of paradigm to take advantage of the HORNET platform each service would have to be developed in accordance with HORNET concepts and API, thus allowing a VoI-based management of both network flows and service components. To address this issue, HORNET explicitly considers backwards compatibility by design. In fact, HORNET can be adopted either in full as a comprehensive architecture or embedded into an existing system. In the latter case, HORNET would manage the behaviour of existing services and solutions by means of more traditional methodologies such as native IP.

7.3 A Reference Scenario

To better illustrate how HORNET addresses the issues of Fog computing applications, this Section illustrates a Smart City scenario hosting several services: pollution monitoring, traffic monitoring, and plate recognition. Pollution monitoring analyzes environmental data collected from, e.g., CO_2, NO_x, humidity, and temperature sensors to produce a report on pollution levels, then disseminated to citizens in the area. Traffic monitoring analyzes data from traffic cameras and vehicle counting sensors to evaluate the level of traffic congestion in the city; then, it disseminates situation reports to citizens (either pedestrians or drivers) and police officers in the surrounding area. Finally, the plate recognition service monitors the traffic plates, e.g., to make sure that only cars with a specific permit are driving through restricted

areas. Cars not complying with this restriction are automatically reported to the police.

All these services have a long-time/continuous running nature and will thus leverage software components running on Fog nodes deployed and administrated by the municipality. The fair allocation of resources to this set of various and ever-running services represents a challenge. In fact, static allocation of resources, simply based on weights assigned to service types, would be inadequate in this scenario. In fact, in steady state, all the above mentioned services will likely have access to all the resources they require. But this allocation would not make the system responsive to quick changes of the environment context.

For instance, consider the case of a stolen car report, leading to the insertion of a car plate code to look for in the plate recognition service with high priority. In this case, the plate recognition service becomes significantly more important and the VoI of its associated traffic flows increases considerably. Our SDN-based HORNET solution dynamically manages the transport layer of the Fog environment by configuring how traffic flows with higher VoI are dispatched, i.e., by tuning routing tables to reroute such flows towards less loaded links, thus prioritizing relevant flows promoting them from regular data streams (whose messages may be delayed or eventually dropped by adopting an "acceptable lossiness" approach) to high-priority data streams (that have full access to networking resources of municipality Fog nodes), or both.

Furthermore, consider the case of a missing child report, requiring to analyze a huge amount of data (dramatically greater than in the plate recognition service) collected from cameras by applying computationally expensive face recognition algorithms. The effective and efficient provisioning of such a service would require not only the prioritization of associated traffic flows (like in the previous example), but also (and most relevant) the on demand deployment and instantiation of service components close to raw data sources, i.e., cameras. To this purpose, our SDN-based HORNET solution adopts a two-layer approach: at the application layer it selects if, where, and when there is the need of instantiating novel service components, while at the transport layer it prioritizes messages carrying the new service (either its configuration or the service component itself) to quickly activate it. It is worth noting how by applying face recognition algorithms in newly deployed fog devices close to the cameras it is possible to achieve the notable twofold benefit of increasing the scalability of the service (thus reducing its latency) and of decreasing the traffic on the network (since video streams can be locally processed instead of dispatched to a central server running on a Cloud platform).

To this end, traditional management solutions, or even recent SDN ones leveraging the concept of network slicing, are not enough to address all the issues raised

by the above challenging scenarios. Therefore, to tackle the impelling requirements of such scenarios there is the need to develop solutions capable of reallocating networking and computational resources to service components in a rapid, fair, and fine-grained manner, while addressing the issue of modulating the information dissemination substrate to match the current state and application requirements.

7.4 The Holistic pRocessing and NETworking (HORNET) Approach

The *HOlistic pRocessing and NETworking (HORNET)* SDN-based solution aims to address the issues discussed in the previous reference scenario. HORNET integrates and extends some work on the Real Ad-hoc Multi-hop Peer-to-peer (RAMP) [109] and the Sieve, Process, and Forward (SPF) middleware [33], by dynamically managing the deployment and composition of Fog services according to VoI optimization criteria and by dynamically re-configuring end-to-end communications to maximize QoS [110, 111].

As illustrated in Fig. 7.1, HORNET executes its middleware services on top of edge devices (nodes N1-N5 in the figure) capable of hosting information processing tasks and of operating as routers for traversing flows, coordinated by a centralized Fog SDN Controller. At each node, HORNET instantiates and manages information processing and dissemination tasks in an integrated and context-aware way by leveraging a two-layer approach, with a top service layer and a bottom communications layer.

At the *service layer*, the Fog SDN Controller dynamically re / deploys and de / activates service components on fog/edge devices. Service components can be any software module, ranging from shell scripts and OSGi bundles to containers and even virtual machines, of course with different supported capabilities and different performance in terms of deployment latency. To this purpose, the Fog SDN Controller analyzes service requirements and the generated VoI to modulate resource allocation, by possibly deploying missing HORNET components when and where needed. At the *communication layer*, the Fog SDN Controller dynamically manages Fog nodes to improve the QoS of packet dispatching not only by computing best paths towards the destination and by tuning forwarding rules on intermediate nodes, but also by adopting an SDN-based MLR approach to select the most proper routing mechanism among different abstraction layers.

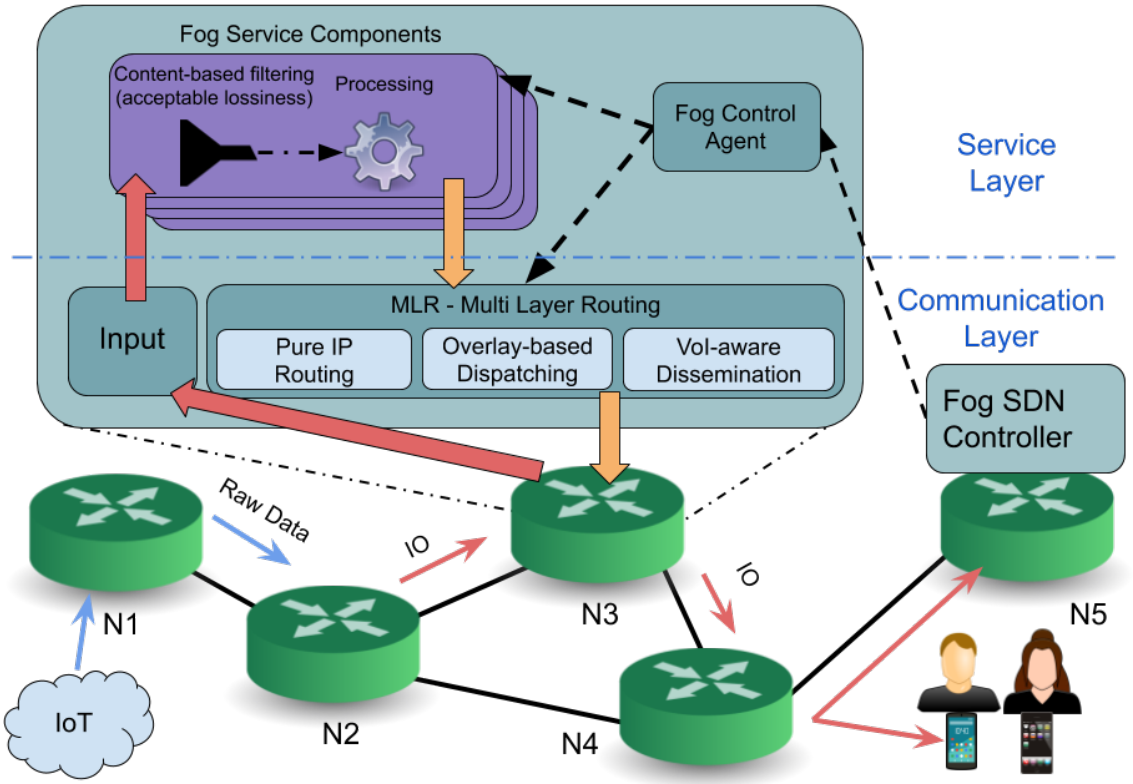


Figure 7.1: The HORNET Concept

7.4.1 SDN-based Multi Layer Routing

This would support the efficient deployment and integration of information-centric services with highly varying fan-in and fan-out there is the need of a communication layer capable of implementing a flexible and context-aware network fabric. To this end, HORNET takes advantage of the MLR-based data plane (in conjunction with the RAMP middleware) supporting three primary routing mechanisms at increasing expressiveness power: pure IP forwarding, (VoI-oblivious) overlay-based dissemination, and VoI-aware overlay-based dissemination.

Pure IP forwarding works by dynamically modifying per-device routing tables with traditional iptables command to reroute vanilla UDP and TCP traffic flows based on IP address destination. This mechanism manages legacy applications in a completely transparent way for edge devices, thus simplifying QoS management.

Overlay-based dissemination supports collaborative packet dispatching among edge devices by adopting routing schemes based on flow ids. In this case, the Fog SDN Controller provides a flow id to applications to label generated traffic with. In addition, the Fog SDN Controller manages Fog nodes by specifying, e.g., that the traffic labeled with a given flow id has higher priority and thus other lower-priority traffic flows should be temporarily delayed.

VoI-aware overlay-based forwarding extends the previous mechanism allowing to tune packet forwarding not only based on the flow id, but also on the VoI carried

by each packet. To this purpose, the Fog SDN Controller manages edge nodes by offering routing rules based on VoI values. In this manner it is possible to support VoI-dependent information dissemination, e.g., by specifying that for a given traffic flow packets carrying data with VoI values below a threshold should be discarded, or differentiating destination nodes based on VoI ranges at the sender as well as on intermediary nodes. In other words, this mechanism dynamically modifies the pipeline of fog services by also permitting the definition of different pipelines for the same data, based on time-varying VoI values. Furthermore, the VoI-aware forwarding also enables a prioritized delivery of service components to Fog nodes, thus allowing a fast instantiation of high VoI service components even in case of a network congestion.

The Fog SDN Controller selectively adopts one or more of the above MLR mechanisms, with per-application granularity, depending on the current state of the network and on the specific application needs. To this end, the Fog SDN Controller would likely adopt Pure IP forwarding or Overlay-based forwarding for legacy applications, while applications built on the top of VoI would adopt VoI-aware overlay-based forwarding. For instance, the Fog SDN Controller can decide that a legacy application with strict low latency requirements (such as video streaming for face recognition [112] or image captioning [113]) should exploit pure IP forwarding, with no overhead due to overlay networking. On the contrary, more articulated applications could be effectively provided only if supported by additional dynamically deployed components, eventually coupled with value-dependent packet dissemination techniques, e.g., by disseminating plate recognition information in a differentiated manner to prioritize newly stolen vehicles.

It is worth noting that while the data plane is based on and enabled by the MLR approach (thus IP native, overlay networking, and VoI-aware dispatching), the control plane is based on the overlay network only. In this manner, information and commands sent to Fog nodes and the Fog SDN Controller can take advantage of the routing flexibility of overlay networking. For instance, in this way it is possible to identify destinations based on a network-independent unique node id rather than an IP address that could change or could be duplicated in different subnets of the same multi-hop Fog environment. In other words, the Fog SDN Controller can dispatch packets related to the control plane independently of how (and whether) routing tables on intermediary Fog nodes have been configured.

The reference scenario presented in Section 7.3 can greatly benefit from the MLR approach adopted by HORNET. For instance, the Fog SDN Controller can exploit MLR to configure the network to reroute low-priority pollution traffic towards limited bandwidth longer paths supported by the overlay network. On the contrary, it can setup IP routing tables on intermediary Fog nodes to forward high-priority

video streams for face recognition towards IP-based short paths.

7.4.2 Value-based Information Processing

The HORNET solution adopts the MLR approach and extends it with the *Adaptive, Information-centric, and Value-based (AIV)* service and information maturity model [36]. On the one hand, AIV proposes an information maturity model that classifies messages in two different categories: raw-data if a message is generated by a sensor and Information Object (IO) if the message is the processing result of a service component. On the other hand, AIV proposes an innovative Fog service model that leverages VoI-based concepts to enable the development of services capable of automatically scaling their resource requirements to their current execution context while preserving high QoE levels.

More specifically, AIV assumes that the processing function of a Fog service emerges as the result of the coordinated orchestration of adaptive and composition-friendly service components, in which each component is responsible to deal with a part of the information processing. This loose definition of Fog services allows to easily support dynamic architectures where 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.). Above this concept, Fog services are defined as a topology of service components connected together with respect to a service description that defines the semantics, the characteristics, and the interactions between service components. For example, the face recognition service mentioned in the reference scenario implements the processing of video feeds collected from nearby cameras (raw-data) through several phases: video transcoding, face detection, and face recognition processing. In this case, a first service component takes care of video transcoding operations by transforming the cameras frame raw-data into IOs that will feed the face detection service component, which in turn passes its output IOs to the face recognition service component to produce valuable IOs for end users interested in consuming them.

It is needless to say that the development of VoI-aware Fog services requires explicit support at the middleware level. In fact, there is the need of VoI evaluation mechanisms of raw data and IOs that are of generic applicability and can be configured to match the needs of the specific service components where the VoI evaluation is used. To this end, the target middleware is SPF, which aims to be the reference implementation of the AIV service model. SPF is available as open source at: `https://github.com/DSG-UniFE/spf` [33].

SPF evaluates the VoI of messages and keeps track of it along their lifecycle. More specifically, the initial VoI associated to an IO m is calculated as a function of

the messages processed for the IO generation and of the priority of the originating service component:

$$VoI_0(m) = SSV(\mathbb{I}(m)) \times FSP(sc(m)) \tag{7.1}$$

where $\mathbb{I}(m)$ is the set of input messages processed for the generation of IO m and $sc(m)$ is the service component that generated m. (Note that $\mathbb{I}(m)$ will typically consist of raw data messages, but might also include IOs, as it is not rare for IoT services to perform information processing at different abstraction levels.) $SSV(\mathbb{I}(m))$, as in "Service Specific Value (calculation)", is a factor that takes into account service-specific considerations when assessing the value of the information extracted from $\mathbb{I}(m)$. $FSP(sc(m))$, as in "Fog Service Priority", is a factor that considers the priority of the service that component sc belongs to, thus assigning higher VoI to the IOs produced by higher priority services. For more details about VoI tracking within SPF refer to [33], and for VoI tracking within a general framework please refer to to Chapters 3 and 5.

Finally, it is worth considering that at the single node level, SPF keeps track of the total VoI of the IOs generated by the service components running in that node. It then uses this information to assign local (computation, storage, and bandwidth) resources to service components according to the VoI they generate.

7.4.3 Optimal processing and networking configuration

HORNET builds on top of SPF to run VoI-based information processing services and leverages its VoI tracking capabilities to tailor the communication layer according to the application requirements and the current VoI they are producing. More specifically, HORNET aims at finding the service component allocation α and network configuration γ which optimize the total VoI delivered to end users:

$$\underset{\alpha,\gamma}{argmax} \sum_{m \in \mathbb{M}(t_n, t_{n+1})} VoI(m, \mathbb{MR}(m)) \tag{7.2}$$

where $\mathbb{M}(t_n, t_{n+1})$ are the messages received by end users within the (t_n, t_{n+1}) time window.

To optimize service component allocation and traffic engineering, HORNET needs up-to-date information about the VoI delivered by Fog services. To this end, SPF continuously monitors service components and, through the local Fog Control Agent, periodically informs the Fog SDN Controller of the total VoI of generated raw data and IOs and if they require additional resources. By using the total VoI associated to raw data and IOs as a resource assignment criterion, the Fog SDN Controller selects the best path for traffic flows and prioritizes the assignment of resources to

services that are providing the highest VoI value to their end users. Furthermore, whenever the Fog SDN Controller assesses that rerouting and prioritization together do not achieve the required QoS (or the generated VoI is extraordinarily high), it can also select to re/deploy new service components.

HORNET uses its knowledge base about the services currently running in the network and the VoI tracking information collected from SPF to optimize service component allocation and traffic engineering at the entire system level. More specifically, HORNET solves the holistic optimization problem in equation (7.2) by leveraging a continuous optimization solution based on an advanced genetic algorithm variant with adaptive mutation. Genetic algorithms are particularly well suited for optimization problems with complex search spaces, because of their remarkable flexibility in chromosome representation, and dynamic aspects, which they are capable of addressing effectively using a varying intensity mutation process (controlled by feedback on convergence speed) [114] and hypermutation triggering to deal with abrupt changes in the system state [115]. As a result, genetic algorithms represent a very good choice for HORNET. However, genetic algorithms are known to suffer from relatively slow convergence rate in some cases. To address that issue, it is worth exploring alternative optimization solutions leveraging Quantum-based Particle Swarm Optimisation and greedy algorithms.

It is worth clarifying that the VoI function in equation 7.2 represents a scalar / univariate quantity. This allows to formulate the resource assignment problem as a single-objective optimization one, and actually represents a significant advantage of the adoption of VoI as an underlying theoretical framework. Alternatively, non VoI-based multi-objective formulations of the resource allocation optimization problems would have required the adoption of a significantly more sophisticated optimization solution, for instance NSGA-II, and most likely less performing from the convergence rate perspective.

The holistic approach to information processing and traffic engineering management adopted by HORNET allows to effectively maximize the overall QoS of the most important service components, i.e., those producing the highest VoI. For example, in case of a stolen vehicle, information about car plates suddenly becomes much more important – hence valuable – for the police. As a result, since the plate recognition service starts delivering higher VoI, the HORNET SDN controller exploits MLR mechanisms at the transport layer by increasing the priority level of packets carrying information about car plates. Once notified about the new priority level, intermediary nodes start dispatching such packets with higher priority as soon as they arrive, but only if the per-packet VoI is high. For instance, a packet carrying a car plate whose picture is blurred (i.e., not useful for the service) has a high priority but low VoI, and thus it can be delayed. On the contrary, intermediary

nodes forward packets related to other services with lower priority only if and when there are no plate packets in the queue. In other words, the dispatching of packets with lower priority is delayed whenever a car plate packet arrives; in case the queue of outgoing lower VoI packets increases too much (and thus packets are delayed for a long period), eventually they can be dropped in an "acceptable lossiness" fashion.

This is even more relevant in the case of missing child report. Since it generates IOs with a very high VoI, the SDN controller exploits the application layer mechanisms to trigger the deployment of new service components in charge of applying face recognition algorithms close to cameras, thus providing more computational resources to process most relevant raw-data. To this purpose, it exploits MLR to provide maximum priority to packets carrying software components to be deployed, temporarily delaying the dispatching of traffic flows related to any service despite their priority and VoI. In this manner, it is possible to achieve the notable benefit of promptly delivering service components also in case of congested network. Then, if the child missing alarm is called off, the Fog SDN Controller sets priority to former levels and stop/decommission service components to release computing and networking resources.

Finally, it is worth mentioning that the VoI-aware MLR approach allows HORNET to decide how to optimize the communication layer during service provisioning, with no need to impose service stops for static re-configurations at service launch time. For instance, the traffic monitoring service typically has a high number of consumers and can be efficiently carried on top of the overlay based dissemination mechanism, also taking advantage of device-to-device (D2D) communications and step-wise efficient multicasting. Furthermore, HORNET can transfer high-VoI video streams for face recognition over high quality paths without any delay and without dropping any packet, while assigning limited network resources to low-VoI video streams for plate recognition that can afford to be slightly delayed or partially dropped ("acceptable lossiness").

7.4.4 Architecture and Implementation Insights

Fig. 7.2 outlines the overall architecture of a Fog service node in HORNET. The depicted components are deployed and activated on each Fog node and allow the node participation to the HORNET Fog environment. In addition, for each Fog environment there is one node acting as the Fog SDN Controller by registering itself to the local RAMP as "Fog SDN Controller" service, thus allowing remote Fog nodes to dynamically discover and register to it. Based on the information provided by remote Fog nodes, the Fog SDN Controller generates a weighted graph representation of the topology.

Delving into finer details of each Fog node, the overall architecture is divided into the Control Plane and the Data Plane. The Control Plane (Fig. 7.2, top) primarily consists of Link Connectivity Manager (LCM) and Control Agent (CA). LCM manages single-hop links and provides network status information. CA gathers information and exploits the overlay network to send data to the SDN controller and receive commands from the SDN controller. More specifically, the Communication sub-component appropriately controls the underlying MLR component to dynamically configure available routing mechanisms, while the Service sub-component dynamically deploys and activates software modules to enrich Fog nodes with additional capabilities required to correctly provide requested services.

The Data Plane (Fig. 7.2, bottom) consists of the RAMP middleware, the MLR component, and the SPF middleware enhanced for Fog service dynamic composition. The RAMP middleware supports the creation of the multi-hop overlay network with best-effort packet dispatching. MLR properly manages packets received by the RAMP middleware on a per-flow basis, by adopting the listener-based Data Plane Forwarder (DPF) to intercept incoming overlay network data packets and apply routing rules related to overlay-based and VoI-aware MLR layers. Furthermore, it is worth noting that Pure IP is logically part of the MLR component (and it is configured by the local SDN CA), but packet forwarding is actually performed by the operating system through kernel routing tables based on the received HORNET indications.

The MLR layer extracts the content (together with VoI metadata if available) of incoming packets of interest to any of the services running on the Fog node and forwards it to SPF, which in turn dispatches it to the concerned service component(s). In the (likely) case the processing leads to the generation of higher level IOs, the latter – along with their VoI metadata – will be forwarded to MLR in charge of selecting the proper communication mechanism and finally dispatching it.

The current implementation of the Fog SDN Controller adopts the Graph Stream library to identify best paths to provide Fog service, e.g., by applying Breadth First or Dijkstra's algorithms based on different cost functions. In particular, when an application requires to the Fog SDN Controller the best path to access a service, it also gets one of the three already developed routing mechanisms: Pure IP, i.e., managing the Fog environment to modify operating system routing tables of intermediary nodes; Overlay-based dispatching, i.e., exploiting the RAMP middleware to forward packets based on a flow id senders have to tag transmitted packets with; and VoI-aware, identifying the path towards the destination based on the dynamically calculated VoI of packets, e.g., to exploit large bandwidth and small latency for packets with high VoI (thus ensuring better QoE) and less capable paths for packets with low VoI.

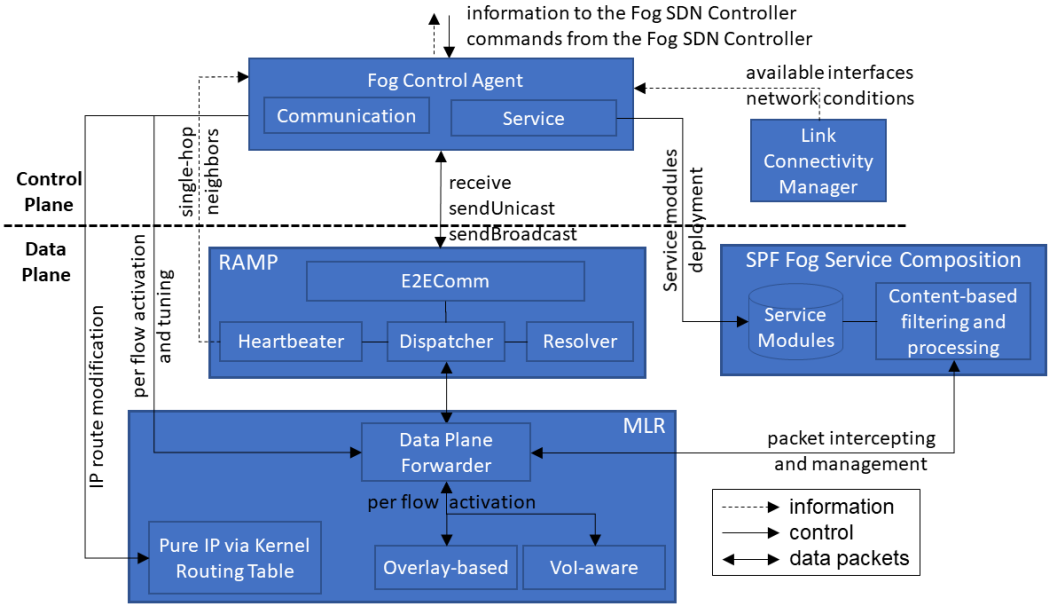


Figure 7.2: Architecture of the HORNET SDN solution.

7.5 Experimental Evaluation

This Section presents the evaluation of the proposed solution with a Java prototype as well as in a simulated scenario, with the twofold objective of demonstrating i) its feasibility and efficiency in a real-world (but small-scale) scenario and ii) its capability of dynamically deploy and and compose service components in a wider simulated environment.

In particular, the purpose of the in-the-field experiments (based on the RAMP middleware, see Section 7.5.1) is to evaluate the performance of MLR-based network reconfiguration by measuring the control plane latency, also considering the challenging case of service component deployment with bandwidth saturation due to high traffic load. Then, performance results achieved with the prototype are used to configure a larger-scale simulated environment (based on the Phileas simulator, see Chapter 4). In this manner, it is possible to faithfully reenact an actual prototype in a simulated environment to evaluate how the proposed solution is able to promptly reconfigure service components (and the network in general) by considering the activation of multiple devices and heterogeneous service instances to optimize the overall VoI.

7.5.1 In-the-field performance evaluation over real testbed

To demonstrate the feasibility over a real world scenario, HORNET is evaluated in a testbed consisting of 5 Raspberry Pi 3 Model B+ connected one another in a kite-like network topology via either Ethernet or IEEE 802.11 and based on a Java

prototype available at `https://github.com/DSG-UniFE/ramp`. It is worth specify that the evaluation of the efficiency of the control plane is based on two primary guidelines: the time required to setup MLR forwarding rules on Fog nodes and the capability of dynamically deploying service components.

The first step is to evaluate the overhead due to rule management by measuring the *rule management cost*, i.e., the time spent from when a node requires to the SDN controller the deployment of a new traffic rule to when the new traffic rule starts to be enforced on Fog nodes. In the case of pure IP, the SDN controller has to setup a new routing rule and send it to a Fog node, in charge of applying it. In case of VoI-aware forwarding, the SDN controller prepares the Java class containing the logic of the rule and sends it to a Fog node, deploying and registering it to the overlay networking RAMP middleware.

To this end, Fig. 7.3 depicts measured traffic rule deployment costs in a regular situation, i.e., without network congestion. In the (native) IP case it takes about 483 ms, while in the VoI-aware case only 59 ms. Such a difference is justified by the fact that in the former case there is the need of modifying routing rules at the operating system layer, which is a time-consuming operation also due to the required context switch and the time required to execute the `iproute2` command. Instead, in the latter case, once the VoI-aware routing rule reaches the interested Fog nodes, the deployment and registration procedures take less time since they are implemented within HORNET directly at the overlay layer (they do not require reconfiguration at the operating system layer). In fact, at the reception of a file representing the rule as a Java class, HORNET stores the file locally, then instantiates it by directly interacting with the Java class loader (specialized to specify custom source directories specifically containing VoI-aware routing rules), and finally registers it in a key-value store with the available VoI-aware rules. Such procedure does not involve any time-consuming operation and can be efficiently done in much less than 100 ms as reported in Fig. 7.3.

On the other hand, Fig. 7.4 shows the qualitative trend related to the delivery throughput of a service component (size of 5 MB) via a congested link (nominal bandwidth of 3000 KB/s). More specifically, at time 1 s a traffic flow carrying data packets starts to fully exploit the available bandwidth. Then, at time 7 s the SDN controller exploits the same link to deliver the service component at maximum priority. Slightly after the new control traffic flow starts, the previous one is inhibited by delaying the dispatching of its packets. Finally, at time 9 s the service component is completely delivered and networking resources provided again to the previous flow. In other words, the proposed solution is able to deliver control messages even if data flows saturate the bandwidth.

Finally, the *rule enforcement cost* metric is evaluated, i.e., the time required to

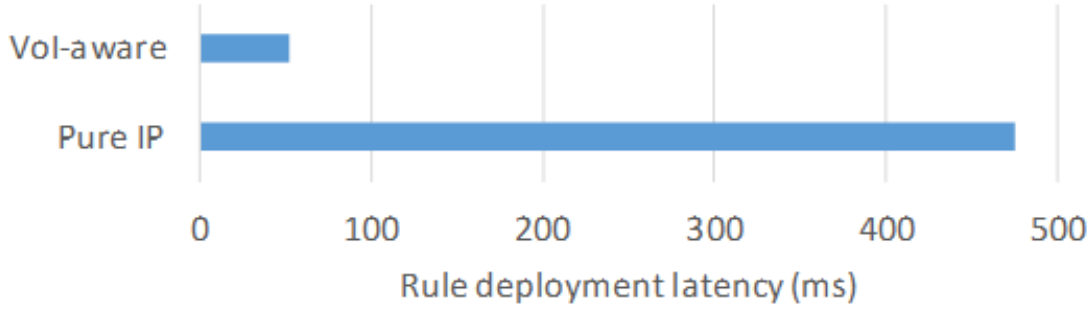


Figure 7.3: Rule management cost.

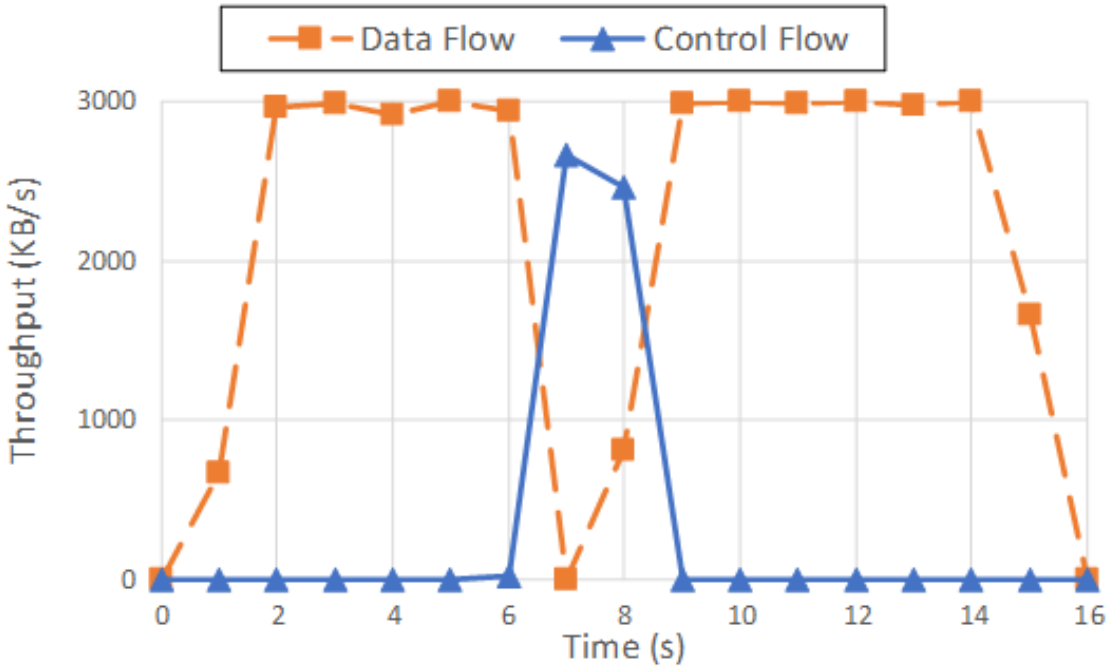


Figure 7.4: Priority management of control and data flows.

enforce rules on intermediary nodes. In this manner, it is possible to better assess the suitability of the proposed solution for challenging use cases requiring the prompt dispatching of packets. To this purpose, the following experiment compares the time required on a Fog node to identify the next node without and with considering the VoI. In the former case, it requires to retrieve the flow id from the packet header and look up the overlay network routing table maintaining <`flow id, next hop`> mappings. In the latter case, there is the non-negligible additional overhead due to payload deserialization, required to retrieve the packet content and then compute the VoI.

Fig. 7.5 depicts how the rule enforcement value varies while increasing the payload size (from 100 B to 10 KB) at high packet rate (250 packets per second). In the overlay network case it takes about 50 ms, with limited rise at increasing payload size, mainly due to slightly more complex memory management. In the VoI-aware case it takes from 50 ms for 100 B payload to more than 250 ms for 10 KB payload.

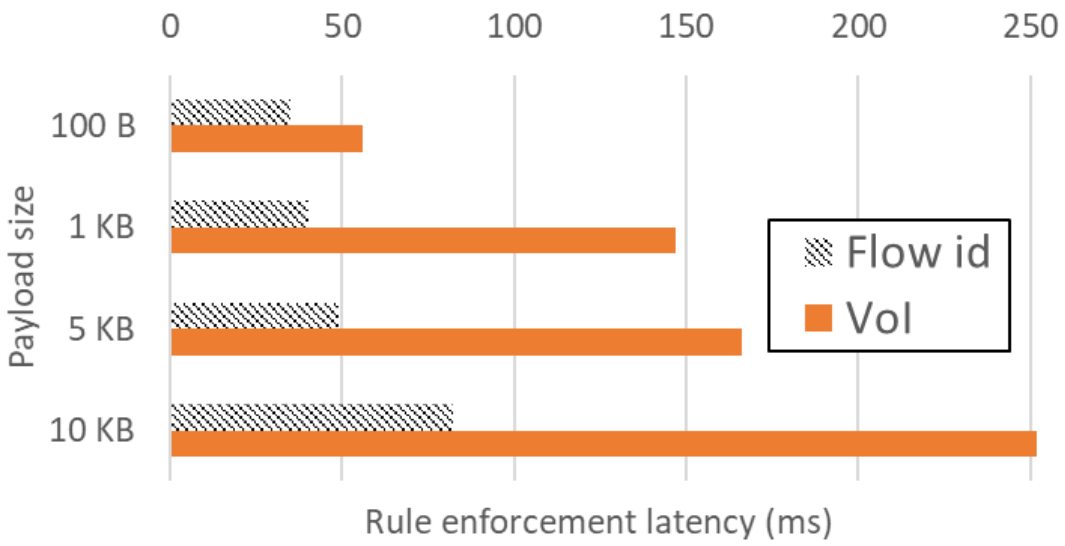


Figure 7.5: Rule enforcement cost at varying payload size (250 packets per second).

In this case the payload size relevantly impacts on the achieved performance, due to the additional time required for deserialization and to inspect the payload content.

Overall, performance results achieved on a real-world testbed composed of Raspberry Pi devices based on a Java prototype demonstrate the feasibility of the proposed solution. In fact, by adopting the proposed SDN-based MLR solution it is possible to remotely deploy new service components in a prompt manner (also in case of bandwidth saturation thanks to the priority-based flow dispatching mechanism), while the time required to enforce rules is limited also in the challenging case it is required to deserialize huge packets.

7.5.2 Simulation of Reference Scenario

To consider larger scale scenario, this Section presents the evaluation of the HORNET framework in a simulated environment with the primary goal of testing the articulated reference scenario discussed in Section 7.3, composed of several devices and different services. In particular, simulated experiments allow to present how the dynamic deployment and activation of services composition, triggered by the time-varying requirements and resources of the target environment, influence the service-specific and overall VoI.

The target use-case

For the purpose of the simulation experiment, as for the other results presented during this Thesis, Phileas is used for the evaluation of the specifically designed Fog Computing target use-case. As depicted in Fig. 7.6, the reference scenario of Section 7.3 is implemented in the downtown area of Washington DC, USA. More specifically, the scenario defines 7 different data sources (4 environmental stations

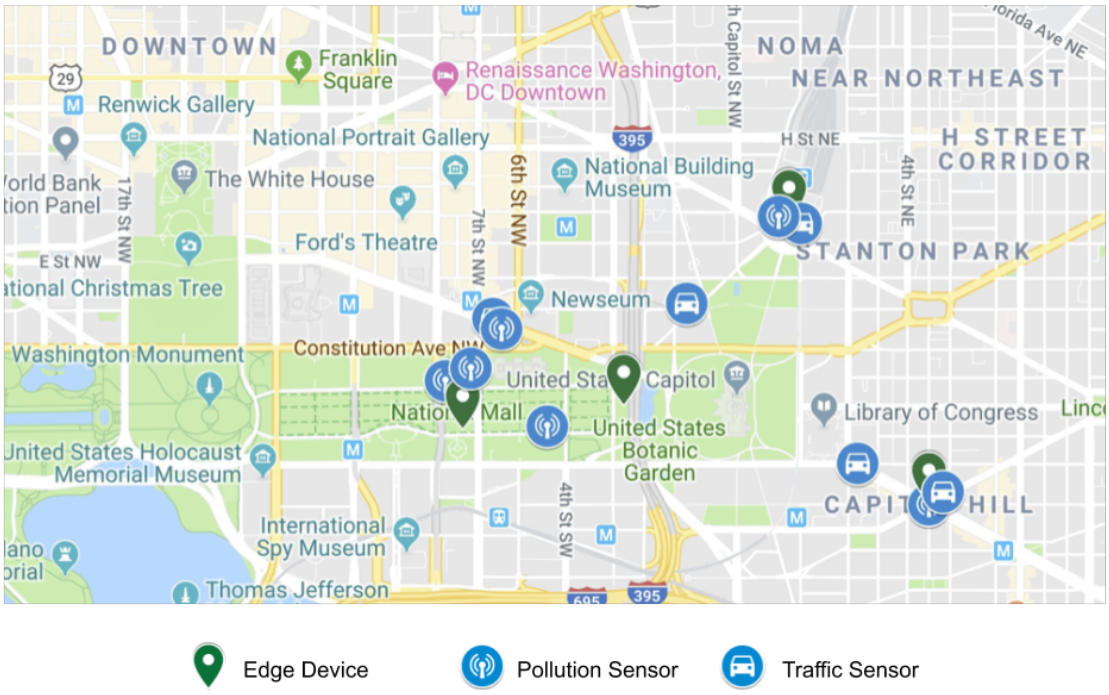


Figure 7.6: Positions of sensors and devices on the map as defined in the scenario.

and 3 traffic cameras) in an area between the National Mall, Capitol Hill, and Union Station. Furthermore, these data sources are configured to generate raw data messages with random and exponentially distributed initial VoI values. The VoI of raw data messages will decay exponentially in both space and time as messages get disseminated. The time between the generation of subsequent messages, i.e., the inter-generation time, and the message size are also random and exponentially distributed.

The raw data collected from IoT sensors serves as an input for the Fog services. At the beginning of the simulation, three services are running on the devices at the edge, analyzing data and disseminating the results in real-time: Traffic Monitoring (TM), Pollution Monitoring (PM), and Plate Recognition (PR). PM collects environmental data from local stations and generates reports with information about the air quality nearby. TM and PR gather video frames from traffic cameras in the downtown area and processes them, respectively, to assess the current viability status and to identify cars with plates of interest.

These services are implemented on top of 10 different components hosted in 4 devices deployed and managed by the municipality. Service components process incoming information according to a lossy and service-specific buffering policy: if they do not have enough resources they will drop messages when the buffer is full. In addition, to mimic the behavior of information processing services, the following experiments rely on the assumption that an IO message m will be generated only

Table 7.1: Characterization of service processing

Name	$\Theta(m)$ distribution	$V_M^{(s)}$	Time decay type, half-life	Space decay type, half-life
TM	$DU(1,10)$	3.5	linear, 1000s	linear, 1km
PM	$DU(1,10)$	3.0	linear, 1000s	linear, 1km
PR	$DU(1,20)$	2.0	linear, 2000s	linear, 1km
FR	$DU(1,10)$	8.0	linear, 800s	linear, 1km

after a random number $\Theta(m)$ of raw data messages, sampled from a probability distribution with service specific parameters, have been received. In the experiment, the discrete uniform distribution $DU(a,b)$ is used to model $\Theta(m)$.

For the purpose of these experiments, equation (7.1) is approximated as shown in eq. (7.3). The initial VoI VoI_0 of an IO message m is obtained by calculating the average VoI of raw data messages RD_i processed to obtain IO m and applying a service component specific multiplier $V_M^{(s)}$ as a weight that takes into account the overall parameters defined in equation (7.1). For the purpose of these experiments, the reference scenario considers linear time and space decay, with service specific half-life parameters. All the parameters considered for service specific modeling are summarized in Table 7.1.

$$VoI_0(m) = V_M^{(s)} * \sum_{i=1}^{\Theta(m)} \frac{VoI(RD_i)}{\Theta(m)} \tag{7.3}$$

VoI-aware service management

The objective of the simulations is to demonstrate that HORNET is able to promptly manage monitored Fog environments by appropriately considering a service components and network re-composition solution that optimizes the overall system VoI. To this purpose, the reference scenario defines 6 user groups interested in the information provided by the services, with the characteristics described in Table 7.2. Groups 1-4 represent a mixture of citizens and municipality officers, while groups 5 and 6 represent police enforcement. Each user group modeled in the scenario has different share interests in receiving information from one of the three services, depending on the type of user and her/his location. While PM and TM provide valuable information for a wider range of users, PR is for municipality/police enforcement use only. Finally, the number of users at time t is sampled from a Gaussian random variable with distribution $N(\mu,\sigma)$.

Phileas is configured to reenact the reference scenario for 24 hours. To model the target use-case, after 4 hours from the beginning of the simulation a missing child report arrives, leading to the immediate activation of the Face Recognition

Table 7.2: Characterization of user groups

ID	Location	Subscriber share TM, PM, PR, FR	Size distribution
1	National Mall, in front of Art sculpture garden	65%, 30%, 30%, 45%	$N(150, 20)$
2	Madison Dr. and 7th St.	70%, 45%, 0%, 45%	$N(90, 20)$
3	Penn. Av. and D St.	60%, 85%, 0%, 35%	$N(115, 20)$
4	Close to Union Station	60%, 30%, 0%, 25%	$N(130, 20$
5	Penn. Av. and D St.	75%, 30%, 75%, 95%	$N(130, 20)$
6	National Mall, in front of Art sculpture garden	75%, 30%, 65%, 95%	$N(130, 20)$

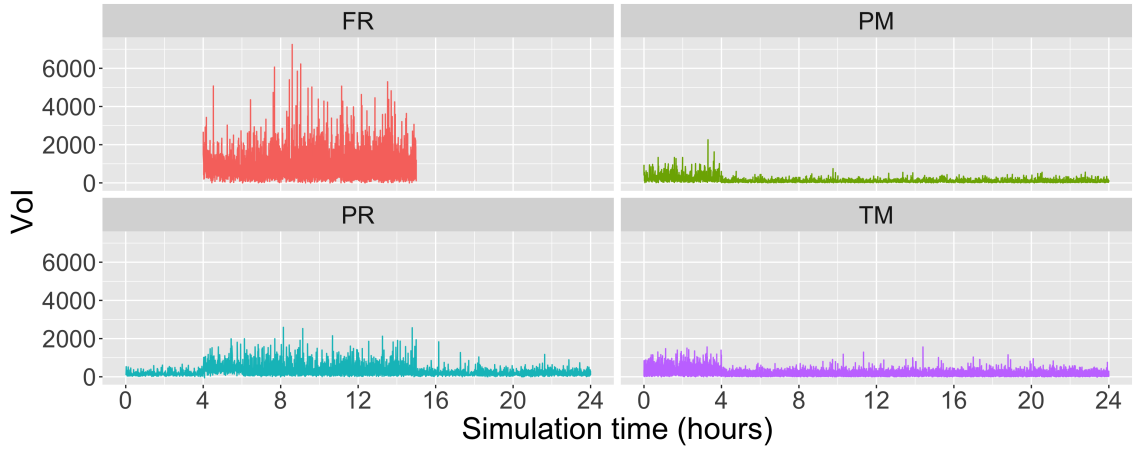


Figure 7.7: VoI produced by each service during the simulation.

(FR) service and of an additional device, as well as of another device one hour later. These changes to the set of service activated and resources available prompted HORNET to activate 5 additional service components (replicas) for FR between 4 and 7 hours from the beginning of the simulation to maximize the total VoI produced. In addition, after 6 hours from the beginning of the simulation, a stolen car report arrives, leading to the increase of the VoI generated by PR (this is simulated by changing the corresponding value of $V_M^{(s)}$ to 6.0) and the activation of an additional fog device from the police. Again, the detection of a change in the VoI produced by PR, as well as of the availability of more resources, prompted HORNET to immediately allocate 2 additional service components (replicas) for PR. Then, to assess how HORNET would respond in case the stolen car and the missing case were found, the reference scenario wil schedule the deactivation of the 3 extra devices and reset the value of $V_M^{(s)}$ for PR between 14 and 17 hours from the beginning of the simulation.

As expected the VoI changes heavily throughout the simulation. Fig. 7.7 depicts with high granularity and at per service level the instantaneous VoI delivered by the IO delivered to users (y axis) versus the corresponding simulation time (x axis). Fig.

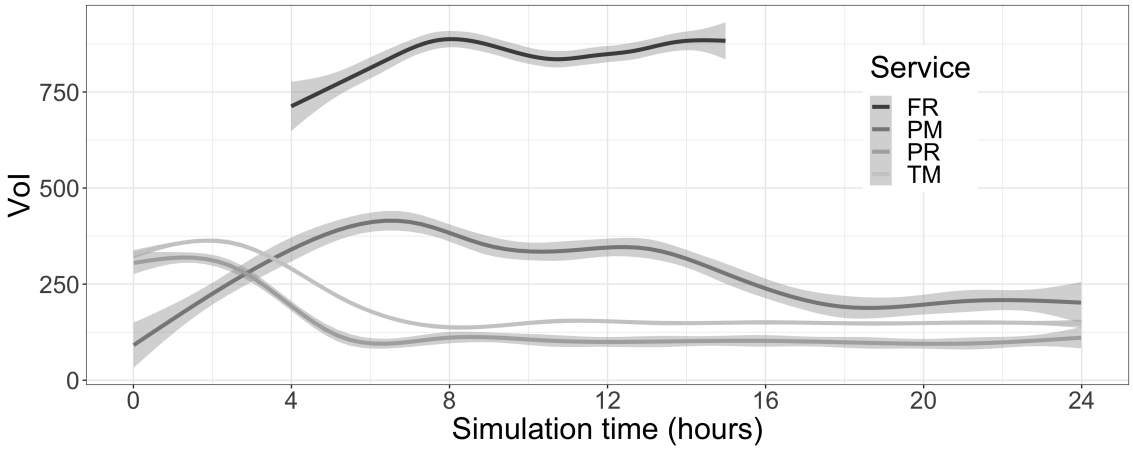


Figure 7.8: Comparison of (smoothed) VoI produced by each service during the simulation.

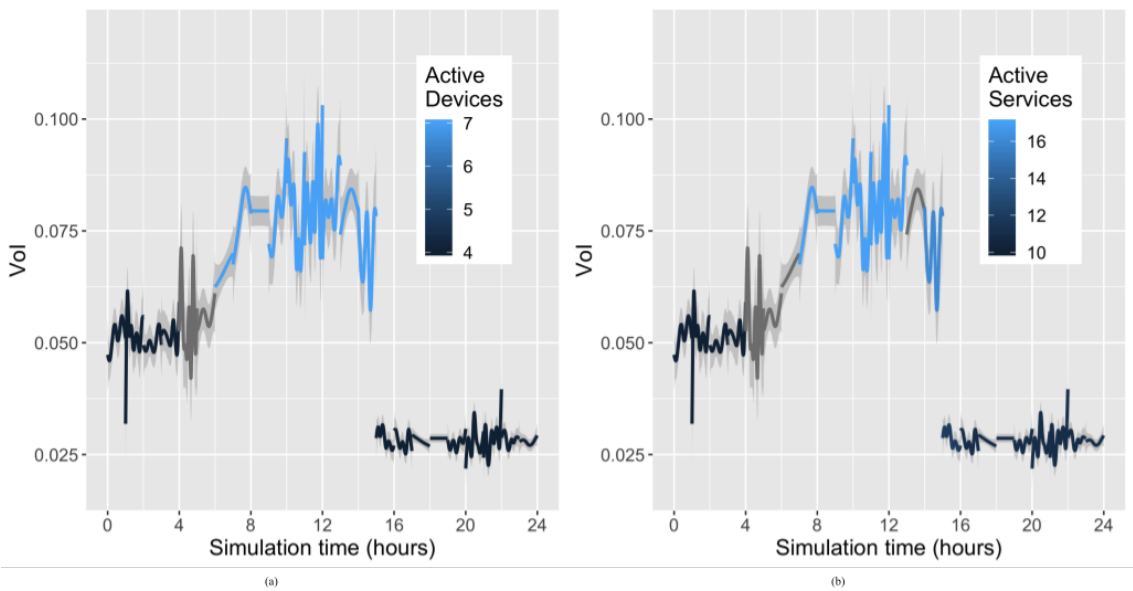


Figure 7.9: Total VoI (scaled) generated during the simulation, in correlation with: a) the number of active services (left), and b) the number of active devices (right).

7.8 presents the same data but in a different form, comparing the curve VoI delivered by each service, smoothed through the interpolation at the entire simulation time level. Both figures clearly depict how the activation of new devices and of the face recognition service impacts the VoI provided to end users of Fog Computing applications. In fact, note how the activation of FR impacts TM and PM in a negative way, even after the activation of new devices. At the same time, the increased VoI of PR after the stolen car report event means that the service receives enough resources to deliver a good amount of VoI. Finally, after the deactivation of FR and of the on demand devices, the VoI curves return to the state at the beginning of the simulation.

Fig. 7.9 provides a different insight on the behaviour of HORNET. It shows the

normalized and interpolated VoI accordingly to a color scale based on the number of active services (Fig. 7.9.a, on the left) and devices (Fig. 7.9.b, on the right). More specifically, the VoI value of each output message are normalized in a $[0, 1]$ range and in 1-hour time interpolation window. The figure clearly shows that a high total VoI is correlated with a high number of active services instantiated and/or devices available for computation, as this allows more valuable information to be disseminated to interested users. This is also confirmed by the data in Fig. 7.10, which shows the number of raw data messages during the simulation. The figure illustrates the number of raw data messages processed every 10 minutes during the simulation, divided per message type. These results demonstrate how the service composition stemming from a triggering-event (the need of locating a missing child) affects the VoI, thus highlighting the framework effectiveness in dynamically (re)allocating resources to optimize the total VoI generated at the system level.

It is worth noting that the computational overhead introduced by the VoI estimation is negligible. In fact, the VoI approach implemented within HORNET allows to selectively filter raw-data messages, thus resulting in a considerably lower amount of information to be processed by service components. Therefore, the computational overhead introduced by the VoI estimation is dramatically lower than the computational resource-saving due to the VoI filtering. To quantify these values, it is worth specify that during the simulation time only 23.6% of the collected video frames and 14.4% of pollution samples were processed.

Overall, results achieved with the Phileas simulator highlight the effectiveness of the integrated VoI-based prioritization at both the service and communication layer performed by HORNET. In fact, by considering the time-varying VoI it is possible to dynamically deploy service components when and where needed, with the positive consequence of increasing the overall VoI itself. In other words, achieved results

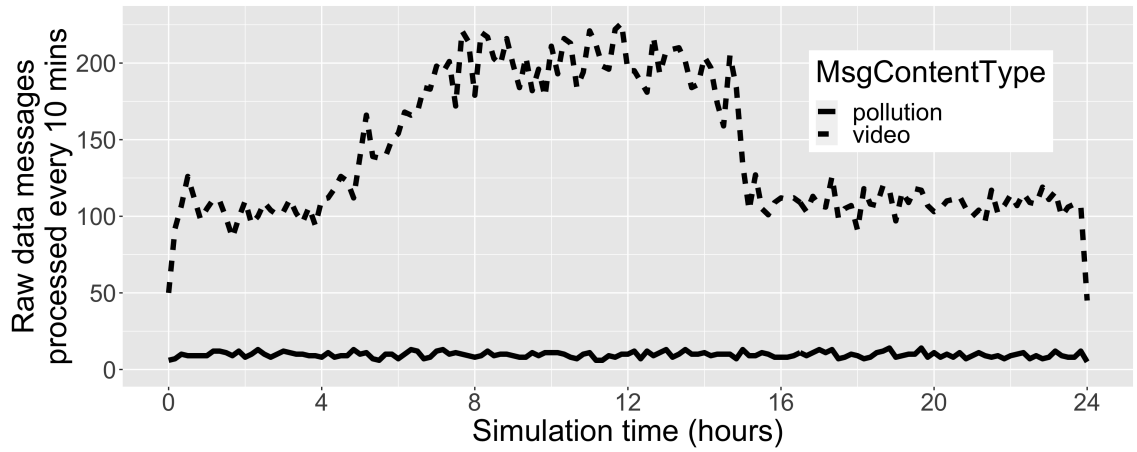


Figure 7.10: Number of raw data messages processed every 10 minutes throughout the simulation.

confirm the capability of HORNET to leverage the availability of more devices to improve the total VoI produced.

Chapter 8

Fog Computing for HADR applications

Humanitarian Assistance and Disaster Relief (HADR) is a particular field in which Fog and Edge Computing can bring their support. Civilian and military personnel involved in the HADR operations can leverage the existing and partly-fault smart city infrastructure to collect useful data concerning the operations. Furthermore, HADR operators can exploit the existing infrastructure for deploying specific HADR services and enabling the dissemination of context-* and location-* aware information.

Within this context, the SPF middleware described in Chapter 4 can be particularly beneficial for managing services and resources operating in a Smart City Infrastructure as demonstrate by [116]. However, to fully explore the capabilities of Smart City assets there is the need to improve the interoperability between military C2 (Command and Control) and core computing systems to support future HADR operations in Smart City environments.

The content of this Chapter is based on the works resulting from the collaboration with the NATO Informations Systems Technology (IST) 147 Research Task Group (RTG) [117, 118], which was formed for investigating the military applications of Internet-of-Things (IoT). As a result of these works, SPF has been investigated as a platform for managing the orchestration of HADR services at the edge of the network. Furthermore, to fulfill the needs of HADR operators, this Chapter presents an extended architecture for SPF that would enable enhanced interoperability between military and civilian infrastructures.

8.1 Smart Cities for HADR Operations

This Section provides a description of a Smart City scenario from a technology perspective and discusses the possible requirements that HADR operators would facing in interacting with the existing infrastructure. In particular, the aims of this Section are to provide a brief overview of Smart Cities and to introduce HADR operations in Smart Cities.

As described along this Thesis, Smart Cities are interesting venues for Fog Computing solutions. The concept of Smart Cities tries to leverage modern ICT technologies to provide valuable services to its citizens. IoT assets deployed in Smart Cities enable the city administrations to remotely monitor, manage and control city activities and services, and create new insights and actionable information from massive streams of real-time data [119][120]. These IoT devices are remotely deployed on buildings, streets, people's houses, industrial installations etc. inside a city. They gather a plethora of information and actuate resources based on demands. Sending all the data to a central server or cloud in raw form puts a severe strain on resources such as bandwidth usage, processing and analysis resources of the core components of the city's ICT, security [121] implications of data transmission and interoperability concerns of the data transmitted from multiple sources. In order to circumvent these issues, solutions such as Edge Computing, Fog Computing, Multi-access Edge Computing (MEC), mobile cloud computing etc. have come into the picture which offload the computation tasks away from the core of the city's ICT systems [122]. This would allow for real-time analysis of the data collected at the edge, filter the data required to reported to the core ICT and allow quicker responses for the operational demands.

Now, with the growth of cities, the tangible resources in the city have increased by many fold. The human population concentrates in the cities now rather than rural or sub-urban areas. This upsurge of population creates a huge demand on city administrations to manage people and its resources [123]. Smart Cities with their ICT and IoT systems try to solve this issue. The natural disasters due to global warming as well as man-made situations like fatal industrial accidents, terrorist attacks etc. call for cities to be prepared for large-scale Humanitarian Assistance and Disaster Recovery (HADR) operations [124]. These HADR operations would require large involvement of technology assets along with human assets.

In this case, IoT assets can provide real-time Situational Awareness (SA) by sensing, computation and actuation since they operate closest to the ground. They are built for the purpose of running on restricted resources such as power which allows them to operate in adverse conditions. Thus deploying Edge Computing solutions on these edge (IoT) assets would inherit the advantages of IoT devices and

further their usage in restricted conditions by reporting filtered out, analyzed and useful data required for HADR operations.

However, *exploiting those assets for HADR operations* is not a trivial task. For instance, there is the need to discover these assets, to interact with them in a automated manner (interoperable API and data-format) for allowing rescuers to exploit the existing infrastructure. Furthermore, these HADR operations are likely to involve the collaboration of both civilian and military personnel, thus requiring an accurate and a secure information exchange. Within HADR operations, military can provide useful services and human-power to assess the after-disaster conditions caused by natural disasters.

To fully exploit the IoT assets there is the need of interoperable solutions capable of dealing with the software and hardware heterogeneity of IoT those assets. Furthermore, there is the need for platforms supporting a quick and easy instantation of software components. Looking into this direction is the NATO Informations Systems Technology (IST) 147 Research Task Group (RTG), which was formed for investigating the military applications of Internet-of-Things (IoT). To this end, the group is looking towards using the SPF platform as middleware for service management during HADR operations.

The following Sections discuss how to enable a HADR full support, by proposing an extended architecture for SPF that aims at the ready deployment by the military as well as the civilian counterparts services for HADR operations. This extension is to improve the interoperability and to enlarge the SPF's range of applications.

8.2 Integrating SPF within a Military C2 Infrastructure

Considering the role of stakeholders envisioned for the SPF applications, the military C2 Infrastructure is one of the possible users of the SPF platform. Especially for HADR operations, where the NATO IST-147 group is looking towards applying the IoT technologies, SPF is considered as a possible candidate for Edge Computing solutions. For HADR operations in Smart City environments, the SPF solution can be used to gain hybrid SA i.e. from the IoT assets deployed by Smart Cities as well as by the military at the edge [125, 126].

In the envisioned scenario, the various sensing and reporting platforms from the ICT systems of the Smart City and the military gather SA data about various events and incidents in a disaster or emergency situation. These ICT systems may gather data from various sensors, cameras, human-source intelligence, Non-Government Organizations (NGOs) and first-emergency responders to help citizens across the

city. They might get and report specific inputs which might of interest for further analysis and operations to help citizens or secure resources in the emergency.

Based on the data or reports gathered form the ICT systems, further analysis might be performed to find out what action needs to be performed correspondingly. These might be according to the operational needs as set for the disaster recovery situation where a specific set of actions need to performed to assist citizens in that situation. For example, an incident might be reported by a team located on ground of a severe fire in a building and based on the severity of fire, the city's ICT systems might want to direct the nearest located fire trucks to the rescue area to save time and increase efficiency of the rescue operation.

After getting the analyzed data from the ICT systems, the administrators (Smart City) or commanders (Military) might want to initiate an action to work on the received inputs or intelligence. These can be autonomously done through the C2 applications to initiate an action. For example, whenever the C2 application knows about a fire in a building, based on the camera resources available at the edge, it would trigger the cameras at the vicinity of the incident to send in the video streams to monitor the situation. Also, manually a resource can be chosen to take some action. For example, the map being displayed by the C2 UI can be used to pin-point camera resources on the ground and activate them to report the video streams. These C2 applications can be running on the central command's ICT systems or on a Mobile Tactical Operations Center (MTOC) based on Federated Missions Networking (FMN) architecture.

The demonstration at the International Conference on Military Communications and Information Systems (ICMCIS 2018), Warsaw, Poland, 22-23 May, 2018 by the IST-147 group, showed the possible implementations and use-cases for military applications of IoT and military C2 systems [125]. These demonstrations proved that SPF is particularly well suited as on edge computing platform showing use-cases for running face recognition and object detection algorithms [127] for video captured at the edge and streamed back to the C2 end. However, in order to highlight its potential within military environments, the Warsaw demo identified the need to extend the SPF architecture to support better integration for military systems and processes.

From the management perspective, there is the need to develop low interfaces with external C2 applications such as Android Tactical Assault Kit (ATAK) [125].

<Org_Id >/<Country_id>/<ownership_type>/<device_class>/<device_type>/<message_type>/<data_type>/"JSON_msg"

MQTT Topic: NATO/GER/Private/Requestor/C2/Command/CameraFeed/"JSON_msg"
JSON Payload: {"Obj_id": GerC2, "lat": 52.245621, "UTC": "2018-02-01 10:17:30.227125", "lon": 21.012371, "value": HDFeed}

Figure 8.1: MQTT Topic Format and Sample JSON Message

The current implementation of SPF components i.e. Controller, and PIGs is based on proprietary interfaces that could be improved to increase the possible use-case scenarios. These components right now are not abstract in nature and are designed (code-wise) to carry out specific tasks. As a result, whenever a new interaction or operation involving any of these components is required then an entire new functionality has to be added to the existing workflow. Also, future additions and configuration of specific modules would require a lot of rework and thus a lot of redesign of the existing implementation that would satisfy that particular use-cases and might not be applicable for future use-cases.

At the other end, the components in charge of executing SPF services (pipelines and services) right now communicate using a proprietary protocol based on unicast UDP, and disseminate information using the ACM DSPro middleware. In order to better highlight the SPF potentials for military applications, these interfaces should be compatible with IoT-specific along with other legacy protocols. With this regard, the NATO IST-147 RTG identified the Message Query Telemetry Transport (MQTT) as a protocol for message and telemetry transfer within the IoT domain for military operations [126]. Fig. 8.1 shows the MQTT topic format and a sample JSON message being exchanged between the military systems as demonstrated in the Warsaw demo for NATO IST 147 RTG. The interfaces between components and information producers/consumers should adhere to these MQTT message formats for systems to exchange data between themselves, initiate requests and responses, and trigger specific strategies and pipelines at run-time.

Finally, from the execution model perspective there is the need to consider additional methods for running services within a PIG. The services on PIGs are invoked through the processing strategies and pipelines running on them. In fact, the Warsaw's demo highlighted the need to consider different services, such as the transcoding of video formats, using COTS components. At the same time, it is possible to take this opportunity to allow the execution of components based on industry standard protocols, such as OSGi.

8.3 Extended Architecture for SPF

To address these issues, this Section presents an extended architecture for SPF. The extended architecture is designed to make interactions between the components open and extensible for various use-case scenarios or implementations, to re-utilize certain components, and to reduce the coupling and dependencies between the components by adopting common format for data exchange and interaction based on the existing military C2 platform architecture and ICT systems.

Fig. 8.2 shows the extended architecture for utilizing the SPF platform in the

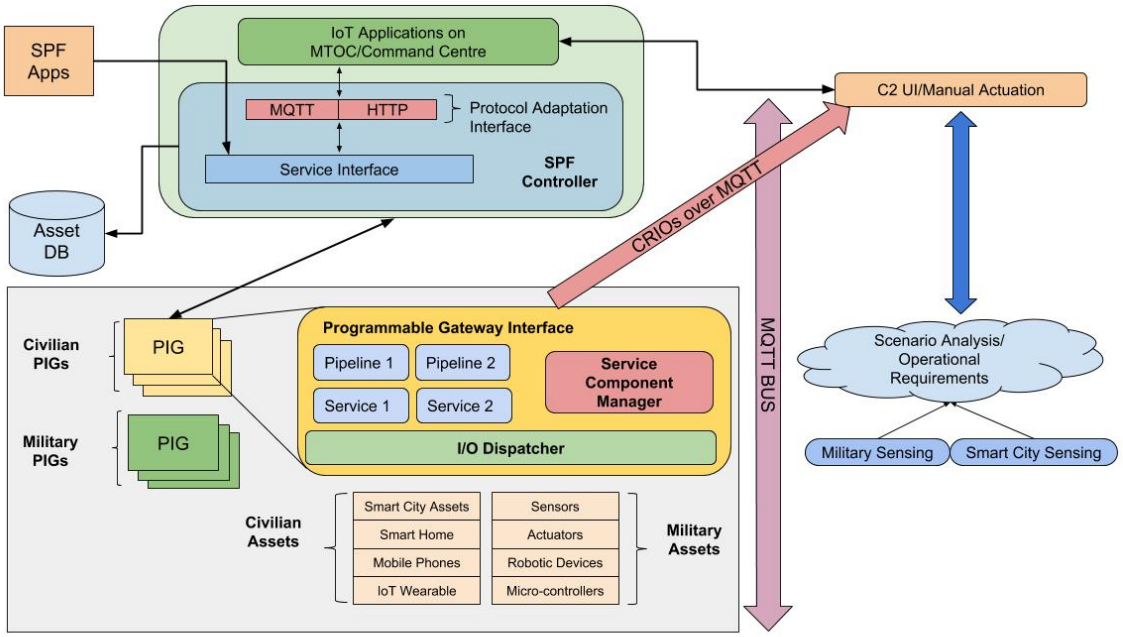


Figure 8.2: Extended SPF Architecture

context of a HADR scenario involving military and civilian ICT systems and assets. The MTOC system houses the IoT Applications for C2 and resource management of IoT resources. These applications and resources employ the SPF Controller for application management and coordination. In this architecture, the SPF Controller issues commands and get responses based on commands received from the C2 application. In order to receive commands, the MTOC application would use the Protocol Adaption Interface (PAI), to communicate with the SPF Controller. PAI is a protocol adapter i.e. it can employ any kind of protocol as suited for the use-case, such as MQTT and HTTP as illustrated in Fig. 8.2. The Warsaw demo demonstrated that MQTT well suits the needs of military applications and it can be used as data messaging protocol. By adopting MQTT, PAI can thus receive the JSON payloads from the C2 applications included in the MQTT messages and interpret the request type. While, at the other end, the SPF Controller makes use of the PAI to communicate with the MTOC application.

The SPF Controller has two main components: the above described PAI and the Service Interfaces (SI). According the workflow of this architecture, PAI listens to receive inputs from the IoT application to execute a job/service over the HTTP and MQTT protocol. Instead, SI acts as an interaction-counterpart for the SPF controller and the SPF based applications. In the proposed architecture, the SI would interact with the PAI by receiving commands and initiating responses. The SPF controller would use the SI commands to initiate its internal functionality of triggering the edge resources or the remote PIGs deployed at the edge. The PAI

will translate any commands received by non-SPF applications into a SI specific command.

In detail, this SPF's operation would involve to observe the implemented PAI to receive commands from MTOC application, to translate the requests from the MTOC into specific or use-case based requests, and to apply the application configuration dynamically at run-time instead of maintaining an application specific configuration file. Instead, the SI would be responsible for locating and triggering the use-case based PIGs based on the requests.

The remote PIG running at the edge has a list of hardware resources which it can interact with such as cameras, sensors, actuators etc. The PIG is responsible for triggering specific services on the deployed resources to execute use-case specific tasks. For instance, a request from the SPF Controller can ask the PIG to deliver HD Camera stream from its available camera to monitor a building under fire. The PIG can instantiate a service that sends back byte streams of camera feed back to the SPF Controller. In addition, it can also employ security specific mechanisms to handle military and civilian data streams separately. The triggering of services is based on Strategy-Pattern where a specific instance of a service on the PIG can be instantiated based on the request from the C2 application and its interpretation by the SPF controller.

Based on the service and processing strategies, the Service Component Manager (SCM) decides which pipelines and/or services have to be executed/invoked on the PIG. The SCM has a list of available pipelines for the PIG, and thus triggers specific pipelines and/or services to execute specific tasks and return results. Typically, in the original architecture of SPF, pipelines and services were dedicated to the processing of all content of the same type. However, in this extended architecture, pipelines and services for the Civilian assets and Military assets are differentiated and initiated as separate, since military assets are deployed by the military and are presumed to be more reliable and trustworthy than the civilian assets for which the owner could not be verified in HADR operations.

PIGs can collect data from assets using two different methodologies depending on the type and the capabilities of those assets. First, PIG can periodically polling data from assets in case they expose a REST fashion API, such as traffic cameras located on the street. On the other hand, we also specify that PIGs can collect data by means of the MQTT Bus illustrated in Fig. 8.2, which listens for data sent by assets on pre-defined and configurable MQTT topics.

Apart from sending back the results to the SPF controller running at the MTOC level, the PIG can also send data directly to the C2 application, based on the specific inputs from the MTOC IoT applications. As illustrated in Fig. 8.2 these results/packets (Consumer Ready Information Objects in Fig. 8.2) are delivered as

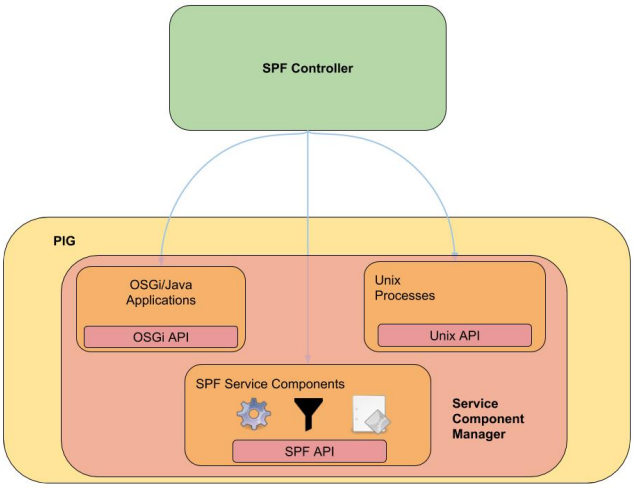


Figure 8.3: SPF: PIG applications

JSON payloads over MQTT. The PIG also employs a proprietary UDP connector which can be used for sending out UDP packets instead of using MQTT in specific use-cases such as delivering video streams. More in detail, remote PIG's operations would involve: to receive and parse commands from the SPF Controller, to configure dynamically the PIG components at run-time instead of using static configuration files, to invoke user-case Service and Processing Strategies based on the request's parameters, and to locate the appropriate pipeline to execute the service on.

Based on the civilian or military edge system at which the PIG is deployed, different and specific pipelines can be deployed for the specific assets. For example, in a HADR scenario that involves the use of civilian assets, video streams provided by cameras deployed on the streets can be used as input data for face recognition pipelines running on PIGs. In this case, the video stream will be elaborated and analyzed to provide to the SPF Controller and the requesting C2 application the results of the face recognition algorithm. As illustrated in Fig. 8.2 other examples of assets can be sensors, actuators, and IoT Wearable devices located across the Smart City scenario.

8.4 SPF: Applications Deployment on PIG

Another architectural change would involve to make the SPF platform capable of supporting dynamic services instantiation on the remote PIGs at run-time. This architectural change will let the SPF Controller also to upload new applications on PIGs and to schedule their execution and activation without re-configuring or restarting the PIGs. Furthermore, as illustrated in Fig. 8.3 to extend the range of possible services deployable on PIGs to there different types of services running on the SPF platform: generic (Unix) processes, SPF specific applications, and OSGi

bundles.

First, SPF service components are algorithms and tasks specifically designed as integrated services for SPF. These applications are written as part of PIG's software or could be standalone Ruby classes and represent the applications part of the legacy SPF architecture. Furthermore, in the SPF architecture, a PIG is supposed to be executed on a Unix-like operating systems, and thus enabling the execution of architecture-specific applications such as video decoder and transcoder, elaboration tools, and so on. For security reasons, each application is supposed to be pre-installed on PIGs and it needs to be trusted and known otherwise, the execution of an untrusted application could compromise the security of the whole system.

Finally, the adoption of the OSGi [128] [129] specification[1] would support the execution, the composition, and the management of bundles (Java based applications) at run-time. In particular, adopting OSGi enables the possibility to efficient manage applications through containerization techniques, which will let new bundles to be created, uploaded, and activated on request without performing any cold or warm restarts of the other applications running on the same remote gateway.

8.5 Processing a HADR operation within the proposed architecture

To illustrate the capabilities provided by the extended architecture in an HADR scenario, it is worth considering the following use case. In the envisioned architecture, the SPF Controller is assumed to be running on the MTOC or to be a component of an IoT application running on the command vehicle, where the command center of the HADR operations is located. The assumption is due to the fact that the Controller is expected to have larger resources for handling requests from service consumers and notably more stable (not susceptible to crashes due to lack of resources like memory, computing power etc.). The SPF Controller would also be associated with a database containing a registry or records of the PIG resources and available services, so would need larger computation resources as opposed to remote IoT assets/devices.

Fig. 8.4 depicts the workflow for a service requested in the proposed architecture. An example of a possible use-case for HADR operations could be the request to elaborate a video feed from a camera in a certain location, e.g. to monitor the presence of moving objects after an earthquake. This would involve the activation of an object tracking algorithm on the video feed, thus delivering an enriched feed displaying bounding boxes around the moving objects. In this example, a C2 appli-

[1] https://www.osgi.org/developer/architecture/

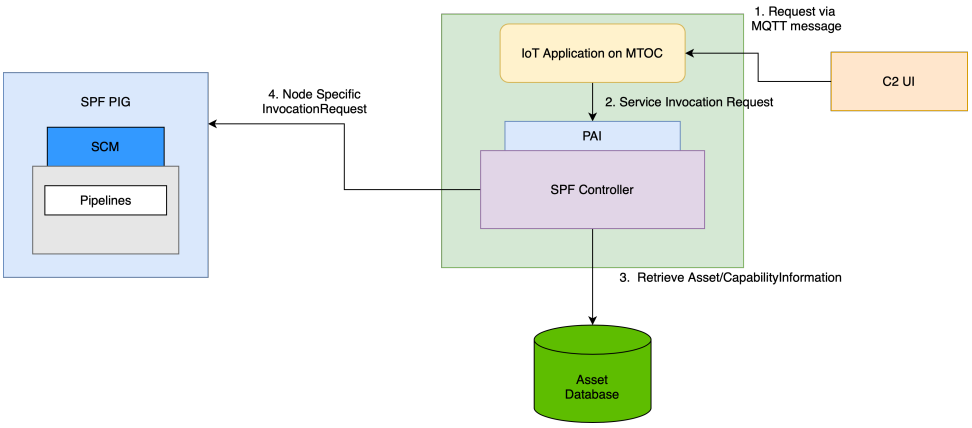


Figure 8.4: SPF: PIG applications

cation or a user would request the elaborated video stream by sending a command to the MTOC application using a MQTT message containing a JSON payload with a list of parameters: the action to be performed (object tracking on video feed), the location of the camera specified as latitude and longitude, the resolution of the requested video feed, and so on.

The request for service is elaborated when the MTOC running the IoT application receives the command and reads the JSON payload contained in the MQTT message. Then, the MTOC sends the service invocation request along with other required parameters to the PAI of the SPF Controller using one of the supported protocol, in this case MQTT. The request is then received by the SPF Controller, which looks up in the associated Asset database to find information regarding PIGs and resources that need to be invoked to serve the received request. In particular, the SPF controller looks for a PIG capable to serve request based on multiple characteristics such as its location (being matched as requested by the C2 application) and the resources (assets, computational power, and so on) associated to that PIG.

When a PIG capable of serving the request is found, the SPF Controller forwards the request to the PIG using the SPF proprietary protocol. Then, SCM schedule the request to start the object tracking algorithm on the video stream on the correct pipeline. If the elaboration requires multiple capabilities such as the transcoding and decoding of the video stream, the SCM would coordinate the effort of more pipelines and/or services. Finally, the elaborated video stream can be sent over the network using the UDP connector or delivered to back to the SPF Controller and then to the C2 application, which requested it.

Chapter 9

Enabling HADR in Smart Cities

After having discussed an extended architecture for the SPF platform tailored for HADR operations, this Chapter takes a deeper investigation on how to facilitate HADR operations in Smart Cities. In particular, this Chapter further investigates the concept of Civil-Military Cooperation (CIMIC) as a collaboration of civilian and military personnel for dealing with natural disasters. CIMIC introduces another level of complexity into the management of HADR because it requires to coordinate the efforts of the different entities involved in the operations also from a security perspective.

To respond to such disasters, there is the need for a comprehensive solution capable of providing a full set of functions for HADR operators. This chapter argues that context-aware and location-aware information is essential for providing Situational Awareness to the HADR personnel involved in the operations. With this goal, this Chapter extends the VoI formulation defined in Chapter 3 to support the modeling of locations of interest (domain) and roles in its formulation. This would enable to prioritize the dissemination of mission-critical information over less valuable ones, also between different domains.

Finally, this chapter reports the experience in developing ACESO, a proof-of-concept middleware that aims to enable HADR operations in Smart City environments. Different from the previous Chapter, which provided a detailed description of an extended architecture for the SPF platform, this Chapter adopts a higher and holistic perspective to provide readers useful guidance for developing Smart City middlewares capable to support HADR operations.

9.1 Concepts

Large scale disasters have an unpredictable nature and can cause widespread damage, overwhelming local authorities and requiring external assistance. Military forces can assist in the rescue operations, contributing their Command and Control (C2) and Logistics assets for disaster relief purposes [130, 131]. Civil-Military Cooperation (CIMIC) is essential for ensuring adequate preparedness for dealing with major natural disasters and enabling a rapid and robust Humanitarian Assistance and Disaster Relief (HADR) operations.

In smart cities, rescuers can connect to the smart city infrastructure, discovering undamaged assets and still functioning services that can help to achieve mission objectives and increase situational awareness. Rescuers can bring some additional assets to support rescue operations and federate them with the assets of the other organizations. Emergency relief personnel would immensely benefit from location- and context-aware information provided by the smart city services and tailored to their needs. However, this presents several substantial challenges.

First, even if partially damaged, modern smart city infrastructure has a truly capillary distribution and thus can generate a plethora of information that might overwhelm the scarce bandwidth and processing resources still available after a disaster [132]. Such a deluge of information could be harmful for HADR operations by exposing rescuers to a flood of irrelevant information among which a small amount of useful data is hidden [133].

Providing a suitable middleware for supporting HADR operations is a challenge. Setting up on demand a dedicated information processing infrastructure is not a viable choice, as it would place an excessive burden on emergency forces and delay the first critical stages of the operations. On the other hand, the repurposing of a typical smart city middleware to support HADR operations presents several challenges. Such systems are usually not designed to be resilient and to work in heavily degraded conditions and they lack capabilities to quickly federate with other systems, i.e., provided by civil or military emergency teams.

These challenges could be adequately addressed by *making middleware for smart cities capable by design of operating in HADR conditions*. More specifically, the middleware should be capable of identifying the information relevant to rescuers and survivors and prioritizing its processing and dissemination. Besides, the middleware should implement a *breaking glass* policy that activates in emergency conditions, relaxing access control on resource utilization and enabling resilient communications. Such controlled relaxation enables a federation of new assets and users while preserving robust security mechanisms for information sharing and system-level logging and auditing.

This Chapter presents the proof-of-concept implementation of *information-centric and context-aware* smart city middleware called ACESO. ACESO leverages VoI methodology and tools to prioritize information for processing and dissemination in HADR scenarios. Furthermore, ACESO supports complex federated HADR operations involving civilian and military organizations by extending VoI approach to the multi-domain environments and implementing label-based information routing and security policies.

9.2 CIMIC Operations in Smart Cities

To discuss the importance of HADR is worth considering a scenario in which a smart city has been affected by a natural disaster which caused the collapse of several civilian buildings with many lives at risk. On such occasion, the *immediate action of rescuers*, e.g., civilian and military firefighters and medical personnel, is vital for assisting the survivors. Emergency services must immediately react to the situation and acquire as much relevant information as possible to prepare, organize and execute appropriate response actions.

However, after a natural disaster, many aspects of the urban area can be profoundly changed and thus negatively affect the local forces HADR operations. For example, earthquakes or floods might inflict damages to the city road infrastructure, thus changing the physical connections between areas of the city. Roads could be interrupted by water, the rubble of fallen buildings, or by direct damage on the road surface that obstructs the passage of rescue vehicles. Obstructed roads cause all the city traffic to re-route to the available transportation links, thus overloading any such connection and creating a situation of traffic congestion that further limits the ability of passage of local rescuers toward the areas in danger. Local forces are consequently forced to re-route or use other transports, such as helicopters or boats, to assist citizens in need.

At the same time, the smart city network infrastructure can be severely damaged when a natural disaster occurs [134]. HADR scenarios can involve multiple network link failures because of physical damage to the communication equipment or because of network congestion. Similarly to the road network, when a part of the communication links fails, the remaining links will be often consequently overloaded by the messages that have been re-routed to reach their destination. Furthermore, in the hours immediately following a disaster, the network might register traffic spikes, which are caused by citizens that try to communicate and reconnect to each other, thus further absorbing the limited network resources.

Disasters might turn the smart city network infrastructure into a set of partially or fully isolated sub-networks characterized by limited bandwidth and unreliable

links. However, the network slicing capabilities of the upcoming 5G standard will likely enable smart city services to run in strongly isolated networks (slices), thus limiting access to IoT assets in that slice. At the same time, the dynamic network slicing capabilities of 5G can be helpful to reshape networks and reallocate networking resources to HADR applications in case of needs. Several research studies are investigating these challenges, see [135, 136, 137].

Emergency response teams will increasingly rely on IT services for assistance in the execution of response activities. Their humanitarian mission would be facilitated by an enhanced situational awareness enabled by constant live data feeds coming from the surviving smart city sensing infrastructure and privately owned IoT devices. This extensive infrastructure of IoT devices enables, e.g., monitoring an area using visual information obtained from local traffic and security cameras. Air and temperature sensors could also be used to identify gas leaks and high temperatures, or fires, that can cause dangerous explosions that could put the lives of aid personnel at risk.

These conditions adversely impact the whole IoT infrastructure, especially if the interaction with the Cloud is crucial for enabling access to the collected data. Unavailability of the Internet or cloud connectivity will affect or even obliterate any smart city services that rely on cloud services for real-time availability of environmental data.

The increasing importance of HADR operations is fostering the cooperation of civil and military authorities for rescue purposes. Civil-Military Cooperation (CIMIC) is increasingly being considered an essential enabler for rapid and robust HADR operations. The involvement of military forces in HADR is promising because the military could leverage their Command and Control (C2) and Logistics capabilities to deploy assets to help with the disaster relief activities rapidly.

However, the military does not own or control the sensing IoT infrastructure of modern smart cities. These IoT infrastructures are referred to as grey assets, which are not as trustworthy as blue assets, owned by the military. However, those authorities will likely focus on different aspects of HADR operations, potentially in areas of the city. Also, participants of the HADR operations might have different equipment and security policies that prevent them from freely sharing information. This is, of course, especially true in the case of military forces.

A typical HADR Operational scenario will involve multiple emergency response teams operating in different areas. It is likely that communication services in these areas will be federated and rely on disruption-tolerant IT infrastructure. Bandwidth will be limited and therefore, of premium value. Given multiple objectives and resource scarcity, the information will also need to be tailored to the end recipients. Finally, as military and civilian responders would usually deploy additional sensing,

processing and communication equipment, there is the need to consider federation and interoperability of information security mechanisms.

Services for HADR operations will likely have a *information-centric nature*, relying on analytics that processes the vast amount of data coming from smart city sensing infrastructure and crowdsensing platforms. In turn, due to the harsh conditions caused by natural disasters, the processing of information will likely be performed locally, often relying on edge-based solutions such as Mobile Edge Computing (MEC) [138, 139, 140].

Since most IT services will implement some form of Situation Awareness, locations represent the most crucial aspect of the tailoring process. Rescuers and civilians are served with detailed information about their surrounding areas while information related to distant areas will be more generic or even no provided. For example, civilians contacting services for help should be informed about the availability of specific drugs they need in the subset of opened stores or relief centers.

Emergency response operations in CIMIC scenarios would significantly benefit from middleware solutions enabling *information-centric and location-based IT services* that can tailor their response to the users' requirements. However, to adequately address the needs of citizens as well as of civilian and military rescuers, middleware should implement a specific set of concepts and functions that allow easy access to the data provided by those services.

9.3 Requirements for HADR Middleware

Smart cities represent an extraordinarily heterogeneous and dynamic environment. They are characterized by the pervasive presence of IoT devices that continuously acquire environmental information. The gathered information is dispatched to consumer applications, local services and Cloud through a multitude of different communication channels (LTE/4G, WiFi, fiber-optic, etc.) [141].

In a smart city, middleware represents the backbone for all applications that heavily rely on live data feeds in order to provide information-centric, time-critical, location-, and context-aware IT services to the citizen. Middleware mitigates the complexity of smart city environments and enables the development of information-centric and location-based applications by providing support services, functions, and tools that orchestrate the multitude of IoT resources and simplify the access to environmental information.

In particular, a middleware for smart cities is typically built on top of extremely sophisticated entities designed to decouple the development of IoT services and applications from the management and access control of resources, such as IoT assets and edge devices. Network softwarization has led to the creation of multiple levels

of ownership or administrative control of smart city assets. For instance, a smart city traffic camera management system might be realized as a virtual infrastructure controlled by the city transportation department and built on top of a shared smart city infrastructure administrated by the municipality IT department. Furthermore, the development of IoT and Fog computing technologies stimulated private actors to enter the smart city infrastructure market with their, often proprietary, solutions. This high number of players and stakeholders translates in the enforcement of a large number of different administrative control and access policies for resources, further increasing the environment complexity.

Unfortunately, so far there has been limited attention towards highly resilient designs for smart city middleware, that would not only allow middleware to withstand natural disasters, but that would make it capable of effectively supporting HADR operations in those conditions. This represents a very challenging but equally promising design approach worth being adequately investigated. Timeliness and ease of deployment possibly represent the most critical goals for IT infrastructure in HADR operations. In contrast to setting up from scratch a dedicated middleware infrastructure to support HADR operations, leveraging the emergency capabilities of an already deployed middleware could grant rescuers a prompt initiative in the first, critical, stages of relief operations, and help them to maintain a high operational tempo.

In order to function effectively in HADR conditions, smart city middleware should satisfy some specific requirements. First of all, the middleware should support a *breaking glass* mode that allows switching to an emergency security policy. In this mode, the middleware activates specific functions that: allow resilient communication; relax access control on resource utilization; enable different, civilian and military, organizations to federate their equipment and services; enforce flexible security mechanisms for information sharing; and implement system-level logging and auditing.

As discussed in Section 9.2, in HADR operations, the smart city network is likely to be fragmented in multiple partially or fully isolated sub-networks, for instance, in case of a CIMIC operation involving a military network that for security reasons cannot directly integrate with a civilian one. Since from the geographical perspective, these isolated sub-networks could be partially overlapping, is preferable to use the more appropriate *domain* term when referring to these sub-network environments. A domain identifies not only a specific area of the smart city but also a homogeneous set of assets owned by a corresponding organization which is logically disconnected from the smart city network infrastructure. It is, at the same time, a geographical, a topological (from the networking perspective), and an administrative concept.

The smart city middleware should allow to connect the various domains involved

in HADR operations. To this end, in the likely case of lack of connectivity, rescuers should be able to deploy dedicated *gateway* devices that integrate with the middleware to realize communications between different domains. In this way, the middleware can leverage the connectivity provided by the gateways to realize a federation of domains in which the different middleware instances cooperate for information processing and dissemination purposes.

The middleware should relax access control policies enforced in normal mode to enable rescue teams to leverage all the assets available for sensing and processing, irrespectively of ownership and administrative boundaries. Moreover, the middleware should be capable of withstanding significant damage to the smart city sensing infrastructure – either physical destruction of IoT assets or crashes/disconnections of the corresponding analytic services. In such cases, the problem can be mitigated by locating still standing IoT assets or deploying new devices in the afflicted area, and integrate them in a new analytics pipeline. To this end, the middleware should integrate dedicated asset discovery and integration functions.

Smart cities middleware supporting CIMIC in HADR operations needs to enable a secure sharing of information between civilian and military teams. The traditional network security measures used to separate military and civilian systems can be in many cases replaced or enhanced by implementing application-layer access control. Such application-layer security policy could be based on several attributes, such as affiliation, role, and reference location of a rescuer.

However, is important to take into account every exception in a highly unpredictable and changing environment of the HADR operations. Although the breaking glass policy enables a temporary overriding of the system security policy, it results in evoking more stringent auditing mechanisms, including a mandatory review of the audit log by a human administrator in order to identify and penalise malicious use.

Because in an emergency situation available bandwidth and computational resource would usually be limited, it is essential that middleware identifies the information critical for rescuers and survivors, and prioritizes its processing and dissemination.

Smart cities present an environment with a high density of devices that gather and publish live environmental data [142, 143]. The process of collecting environmental data is indeed very delicate since it can produce a large quantity of unhelpful or filler information that slows down the process of data elaboration and usage by applications. As the information produced will likely be overwhelming, there is the need to identify the most critical information for rescuers and civilians and forward it to them promptly.

Prioritizing relevant information, however, represents a significant challenge,

which requires specific solutions. More specifically, information relevance is not limited to the timeliness in which is acquired from the environment and received by users, but it is also related to the geographic position of users. Information gathered in the proximity of a user is more likely to be relevant and of immediate use rather than information gathered in more distant locations. There is the need to implement both constrained and location-based information dissemination (geofencing) and city-wide coalition level information sharing.

9.4 The ACESO Middleware

To address the challenges discussed in Sections 9.2, 9.3 this Chapter presents ACESO[1], a middleware solution enabling location-aware and information-centric IT services for CIMIC operations.

ACESO leverages *Value of Information (VoI) methodologies and tools for information processing and dissemination* to offer support for the development of location and context-aware services at several levels. To this end, ACESO is composed of different components designed to extensively exploit the effectiveness of VoI or further enable and simplify the design of context-aware services. More specifically, ACESO is capable of disseminating and filtering information based on VoI, scheduling computation resources based on requirements and ensuring data privacy through attribute-based and context-aware security mechanisms. Besides, ACESO also leverages MARGOT [144] to implement a distributed and federated solution for the location-based discovery of *devices and services*.

These capabilities are required to deal with the harsh conditions that characterize HADR operations, such as network, infrastructure, and device failures described in Section 9.2. To allow services to operate in such conditions, ACESO continuously analyzes the information regarding the operating scenario, e.g. the information about available fog devices, to reallocate computational resources between HADR services or to reactivate or migrate services to different nodes when necessary in a dynamic waving fashion. These policies would helping to mitigate the adverse effects due to natural disasters on Smart Cities environments and to realize resilient and capable location- and context-aware services for HADR operations.

Finally, ACESO leverages *attribute-based* and *context-aware* approach to security, such as location-aware security, to ensure that sensitive information does not leak to unauthorized entities. Such approach allows services to specify tailored and highly restrictive access policies to data while enabling the ACESO middleware to support breaking glass policies for an emergency. Breaking-glass policy could allow

[1]Aceso, daughter of Asclepius and Epione, was the Greek goddess of the healing process.

HADR operators to access private resources and information when required for sake of public safety.

9.4.1 VoI as a foundation for Location-Aware Services

To implement location-aware services, ACESO leverages and extends the Value of Information (VoI) based concepts and tools illustrated during this Thesis. For more detail on the VoI formulation for a general framework please refer to Chapter 5. Within this context, VoI represents a particularly interesting theoretical framework for HADR scenarios. In fact, in addition to the efficient use of resources it allows to naturally consider location-aware aspects as wells as to homogenize them with other concerns, such as information timeliness, and to constraint the burden on information processing and dissemination infrastructure. Both these aspects are of utmost importance in HADR applications.

ACESO adopts a VoI estimation and tracking model that improves the design of the model (presented in Chapter 5) to consider the possibility that the content of IOs might refer to multiple locations and the fact that users might be interested in information related to multiple (and potentially distant) locations.

More specifically, it implements VoI estimation and tracking of each IO that flows through the platform. At the moment t_0 of its generation, the initial value $VoI_0(m, t_0)$ of an IO m is estimated by the originating software component, for instance, according to the content that information refers to or to service-specific policies. Estimates can be further refined through a learning process that considers feedback from end-users [22]. The originating software component also labels IO m as containing information about the set $\mathcal{L}_m = \{m_{ol}, \mathcal{L}_1, ..., \mathcal{L}_n\}$ of interest locations for message m, in which m_{ol} represents its originating location and $\mathcal{L}_i$ are locations to which the content of the IO somehow refers to. We need to consider the fact that a message might contain information relevant to a set of locations, either because they are nearby or because they are in a similar emergency condition and would benefit from the content of the message.

As IO m flows through the network, its content will become obsolete and possibly less relevant. In turn, this means that its VoI will decrease over time and space. The value of message m when received at time $t > t_0$ by user u at location l_u will then become:

$$VoI(m, u, t) = VoI_0(m, t_0) \times TRD(u, t_0, t) \times PRD(u, \mathcal{L}_m, \mathcal{L}_u) \tag{9.1}$$

where $TRD(u, t_0, t)$, as in "Timeliness Relevance Decay", is a factor that takes into account the obsolescence of information, and hence the decrease in the VoI of m in the interval between generation time t_0 and csrrent time t, $PRD(u, \mathcal{L}_m, \mathcal{L}_u)$, as

in "Proximity Relevance Decay", is similar to the $TRD(u, t_0, t)$ component, but its objective is to measure the relevance (and loss thereof) of information with respect to the set $\mathcal{L}_u$ of interest locations for user u. This model allows considering multiple locations of interest for a user, either including its current location or not.

Ideally, it would be possible to assign multiple VoI metadata to the same IO, one per each interested user. However, modelling each user and its set of interest in the system would be extremely difficult. To constrain complexity, this formulation assumes that in practice, users will be grouped into roles and that *the corresponding role identifies the set of interest of the user*. As a result, VoI models can be defined for a given *(information type, user role)* tuple. For example, IOs containing video frames from traffic cameras will have different VoI models according to the role of consumers.

While one might be tempted to define PRD as a function of distance from the centroid of the locations in $\mathcal{L}_m$ and $\mathcal{L}_u$ sets, this would essentially implement the "geofencing" of information. Such geofencing would result in forcing information exchange within an enclosed area and hindering its delivery to users that are interested in locations far from their present, such as higher echelon commanders coordinating the emergency response in remote areas. The nature of HADR operations suggests against the adoption of naive geofencing methodology in favor of more sophisticated solutions. To this end, ACESO adopts two different VoI models, i.e., two different sets of PRD and TRD functions, for intra-domain and inter-domain services, respectively.

With regards to the specific VoI model, ACESO allows using a different set of PRD and TRD functions for each service. This allows services to adopt a VoI model that makes them exhibit a strongly localized behaviour, i.e., models in which the value of an Information Object (IO) decreases rapidly as the IO moves away from the originating source.

9.4.2 Inter-domain routing

For inter-domain communication, ACESO integrates VoI tracking within a more generic information labeling framework for the management of information dissemination. MARGOT is responsible for locating other instances of ACESO running in the scenario, thus allowing them to join and realize information exchange between different geographical locations associated with different domains. Therefore, this process is essential to route information between domains and, at the same time, to make sure that information sharing policies are respected.

More specifically, each instance of ACESO exchanges information regarding its operating domain to build a global status of the location of interests. This status

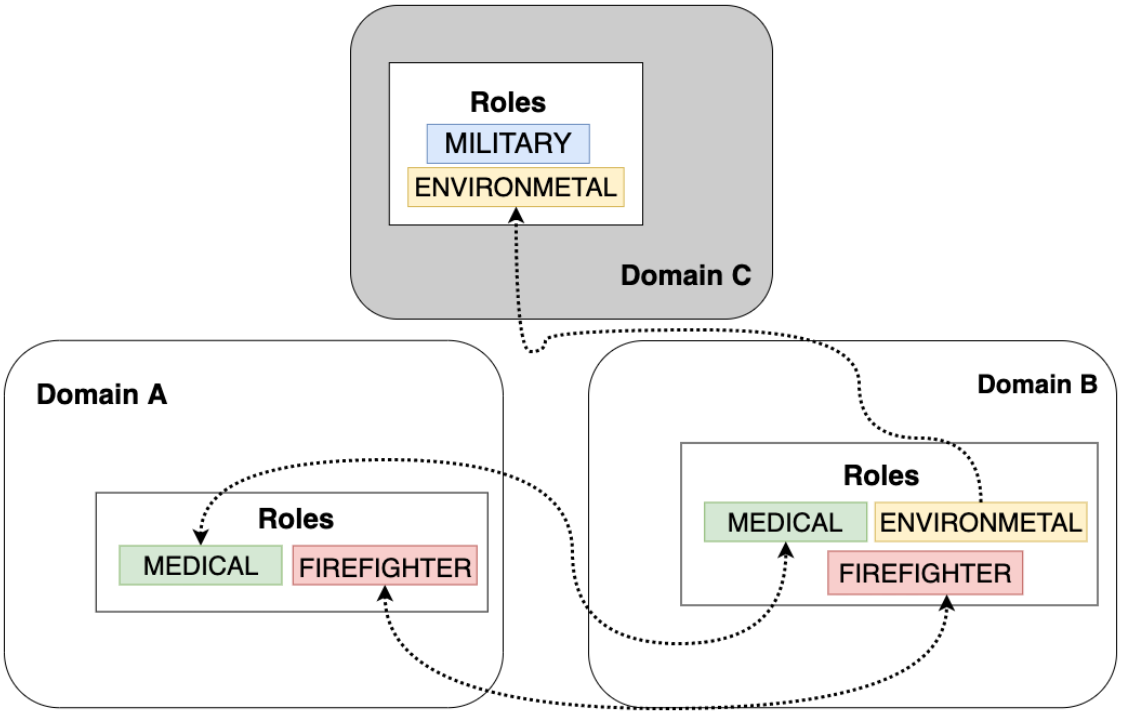


Figure 9.1: Role- and value- based inter-domain routing.

contains a set of rules that associates users and their roles with the locations in which they are operating. These rules are modeled as described in eq. (9.1) and allows each instance of ACESO to be aware of the roles of users operating in other domains.

To realize inter-domain information exchange, ACESO evaluates each generated IO for forwarding by estimating its VoI using eq. (9.1) and the labels the IO is assigned. Let us stress that, for inter-domain communications, ACESO adopts different sets of TRD and PRD functions, in which the decay profile the information is subjected to over space and time is lower than for intra-domain communications. In fact, for inter-domain communications, this evaluation is highly dependent on the $\mathcal{L}_m$ component of PRD function. Therefore, when an IO is evaluated, if there is a matching location $\mathcal{L}_i \in \mathcal{L}_u$ resulting in a non-zero value of the PRD function, the IO is then routed to the instance of ACESO running in location $\mathcal{L}_i$. This matching means that users operating inside location $\mathcal{L}_i$ present a matching role for the forwarded IO.

Fig. 9.1 illustrates multiple domains connected together. More in detail, Fig. 9.1 depicts that users operating in domains A and B share medical and firefighter roles, and consequently, IOs labeled with these roles will be exchanged between domains. In contrast, domain C and domain B share the environmental role, but only IOs originated inside domain B are shared with domain C. Environmental IOs generated within domain C are not exchange with domain B as the security policy prohibits

that information generated inside domain C leaves this domain.

To clarify the concept of inter-domain routing it is worth considering two examples, a first describing service with low TRD and PRD decay profiles and a second highlighting the need for information sharing with no strong emphasis on location. Firstly, a representative example of a service providing information with long-range utility could be a navigation service. IOs regarding the viability status is likely to be forwarded between domains to inform the HADR operators of the overall traffic conditions and to allow them to plan accordingly the optimal route to follow. In this regard, the navigation service will present a lower TRD decay profile. Secondly, a different example is the one in which higher echelon commanders, responsible for coordinating the emergency response, require full access to overall information. In this case, ACESO will forward each IO to the location $\mathcal{L}_u$, where higher echelon commanders are operating, even if $\mathcal{L}_u$ is farther away from the IO's originating location.

Finally, the use of IO labeling allows ACESO to adopt strict rules for inter-domain release of IOs, thus preventing leakage of sensitive information to unauthorized users.

9.4.3 Context-aware security

Information security is one of the main challenges that must be addressed when developing context- and location- aware services. In fact, in order to enable context-aware mechanisms, the nodes' context information needs to be shared with the network or services, thus exposing potentially sensitive data and increasing the attack surface.

However, the context information can also be used to enhance the strength of security mechanisms. One of the most common approaches is to treat the context information as attributes used within *attribute-based access control (ABAC)* policies [145]. For example, the location of a node can be an attribute applicable to both a subject, i.e. entity requesting access, and an object, i.e. information or service that the access is being requested to. Thus, location-aware security mechanisms can be used to enforce dissemination of information originating from one particular area only to users located in another particular area. Moreover, the relative location of a user in respect to an electronic device used for accessing digital information could be used to further enhance security and make sure that a particular terminal is authorized to display information or grant access to a service only if the requesting user is in its immediate vicinity.

Although access control in respect to the IoT information is often with a *read* access - and thus protection of confidentiality of information - it is important to

point out that a *write* access is often even more important, as it contributes to the protection of integrity and availability of information obtained from the IoT devices. In particular, poisoning data sets with false or deliberately crafted information can have far-reaching consequences for correctness and accuracy of operational decisions, potentially leading to insufficient use of resources or endangering the life of personnel.

By leveraging on these concepts, ACESO implements an attribute-based access control (ABAC) manager for the management of the security requirements of HADR operations [146]. Within ACESO, the ABAC manager is implemented as a layer on top of information collection and dissemination components. More specifically, the ABAC manager checks for the integrity of raw-data collected from the IoT devices to avoid poisoned or malicious data to be injected into the processing system. At the same time, the ABAC manager also filters the IOs to be delivered to the end-users of the HADR services by following the security policies defined for the specific scenario.

On the other hand, attribute-based access control is also enforced by the Policy Guards that connect civilian and military domains. This allows a secure sharing of information and avoiding that classified data generated within the military domain would leaked to unauthorized civilian users.

Note that, in some cases, it could be possible to use dedicated network appliances such as *data diodes*, i.e., components that enforce the unidirectional flow of information, which have the advantage of requiring no or minimal modifications to interconnected systems, but also of providing high restriction on supported business processes and operational use cases due to inability to support two-way communication. Slightly more flexible security can be achieved through implementation of two-directional *guard* solutions relying on trusted data labelling for making decisions about releasing - and accepting - information at the network boundary.

9.4.4 Architecture and Implementation

ACESO middleware architecture is illustrated in Fig. 9.2. The middleware is composed of six main modules interacting in a service-oriented fashion. The core module in the ACESO architecture is SPF. Within ACESO, SPF provides functionality to deploy, activate, and configure services on gateways running on fog devices or in Cloud platforms. To do so, SPF collects information about services running on controlled gateways and provides a specific API, which offers mechanisms allowing reconfiguration of computational resources.

The Service Controller module query the status information to continuously monitor the status of running services. On the one hand, in case of a faulty service resulting from the devices' unavailability or network faults, the Service Controller

can generate a new allocation for services using the remaining computational nodes. On the other hand, in case of unresponsive services due to the overloading of computational resources, the Service Controller adopts VoI management policies to deallocate resources from the least valuable applications to effectively redistribute them to more valuable services in the current scenario. These capabilities become crucial when the middleware operates in disaster recovery mode. It is worth specifying that for operating such reconfigurations, the Service Controller can implement the optimization models described in Chapter 5 and Chapter 6.

ACESO defines two information bus for services operating within the middleware. The information bus are designed on the top of VoI methodologies and tools to filter and prioritize data during the phases of acquisition and dissemination. In particular, the VoI Bus evaluate of each raw-data message received from IoT devices within a domain and applies a VoI based message filtering that selects only the most relevant messages. Only filtered raw-data are then processed by the services running on the top of SPF, thus taming the deluge of data generated by the smart city IoT infrastructure. Once the services transform raw-data into ready-to-use information objects (IOs), ACESO enables data dissemination to the end-users by leveraging on the DSPro middleware[147]. DSPro realizes disruption tolerant VoI-based dissemination of information on the top of peer-to-peer architecture, in which each peer acts both as an information provider and consumer. Each DSPro peer periodically exchanges a message called *node context* containing multiple information such as the peer position, role, and range of influence (a geographical area on which the peer is interested in). Thanks to this process, a DSPro peer can decide if an IO should be delivered to another peer or not.

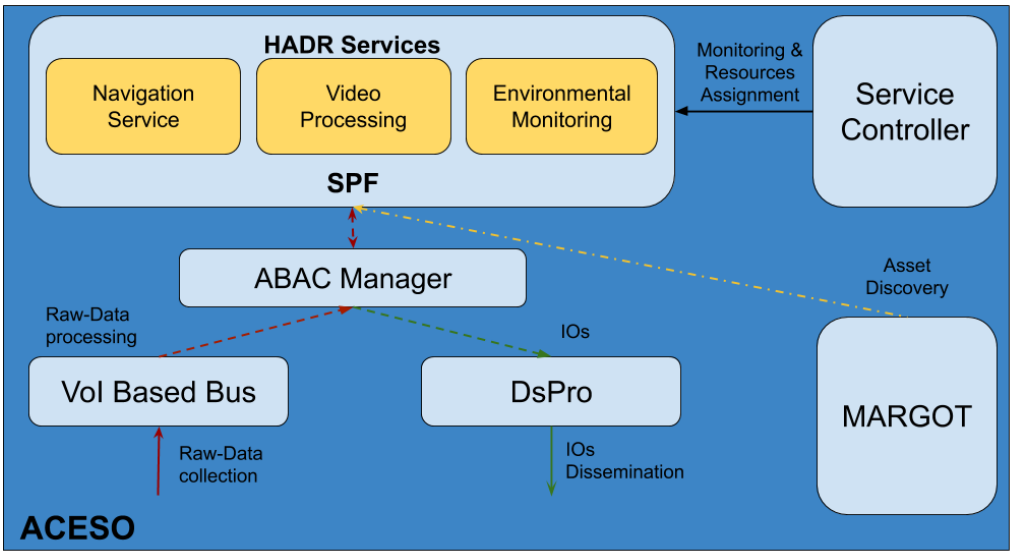


Figure 9.2: The ACESO middleware runnning several HADR rescuers services (in yellow).

The information bus offer an information filtering mechanism based on the context of information and nodes, which usually resolves as an evaluation of standard context attributes such as position, time, application-given attributes. The ABAC manager provides a further filtering mechanism. The ABAC manager is responsible for implementing the attribute-based access control within ACESO. As depicted in Fig. 9.2, it acts as an information filter on two sides. Firstly, it checks raw-data messages to be processed, thus avoiding malicious data to be injected into the processing system. Secondly, based on the operating domain and the defined security policies, it injects to DSPro only the IOs that can be safely disseminated.

Finally, ACESO also integrates MARGOT, an auxiliary service for the discovery of resources within the IoT domain [144]. MARGOT discovers assets and services available within the Smart City infrastructure leveraging proactive-based discovery agents that support multiple communication protocols, e.g. MQTT, CoAP. More specifically, these agents exploit the discovery mechanism of widely used IoT communication protocols to retrieve information without or with minimal prior knowledge of the network.

9.5 Enabling HADR with ACESO

To describe how ACESO would enable effective location- and context-aware services and information dispatching within a HADR operation, this Section presents a fictional HADR scenario illustrated in Fig. 9.3. In particular, Fig. 9.3 shows the collaboration of civilian and military personnel operating in three different domains, illustrated using the grey and blue asset logic: the grey color represents a civilian domain and the blue color a military one.

This HADR scenario introduces a firefighter team departing from the operation center to assist the victims that require immediate assistance. Aside from the firefighters, during HADR operations, also other local forces can be involved. For example, Fig. 9.3 depicts a medical team that operates in symbiosis with the firefighters to provide medical assistance to victims in serious conditions. Completing the scenario are the other teams depicted in Fig. 9.3: a civilian team flying on a helicopter and a military team that assists the local forces by providing them resources, personnel, and hardware.

However, while the main objectives are common to all forces, rescue victims and damage control, each team has different tasks that need to accomplish to achieve the common goals. Firefighters' tasks usually involve the extraction of victims trapped under the fallen buildings or to extinguish fires. On the other hand, the objective of the medical teams is to stabilize the health conditions of the injured civilians and to bring them to the nearest medical care center. Instead, the team flying on the

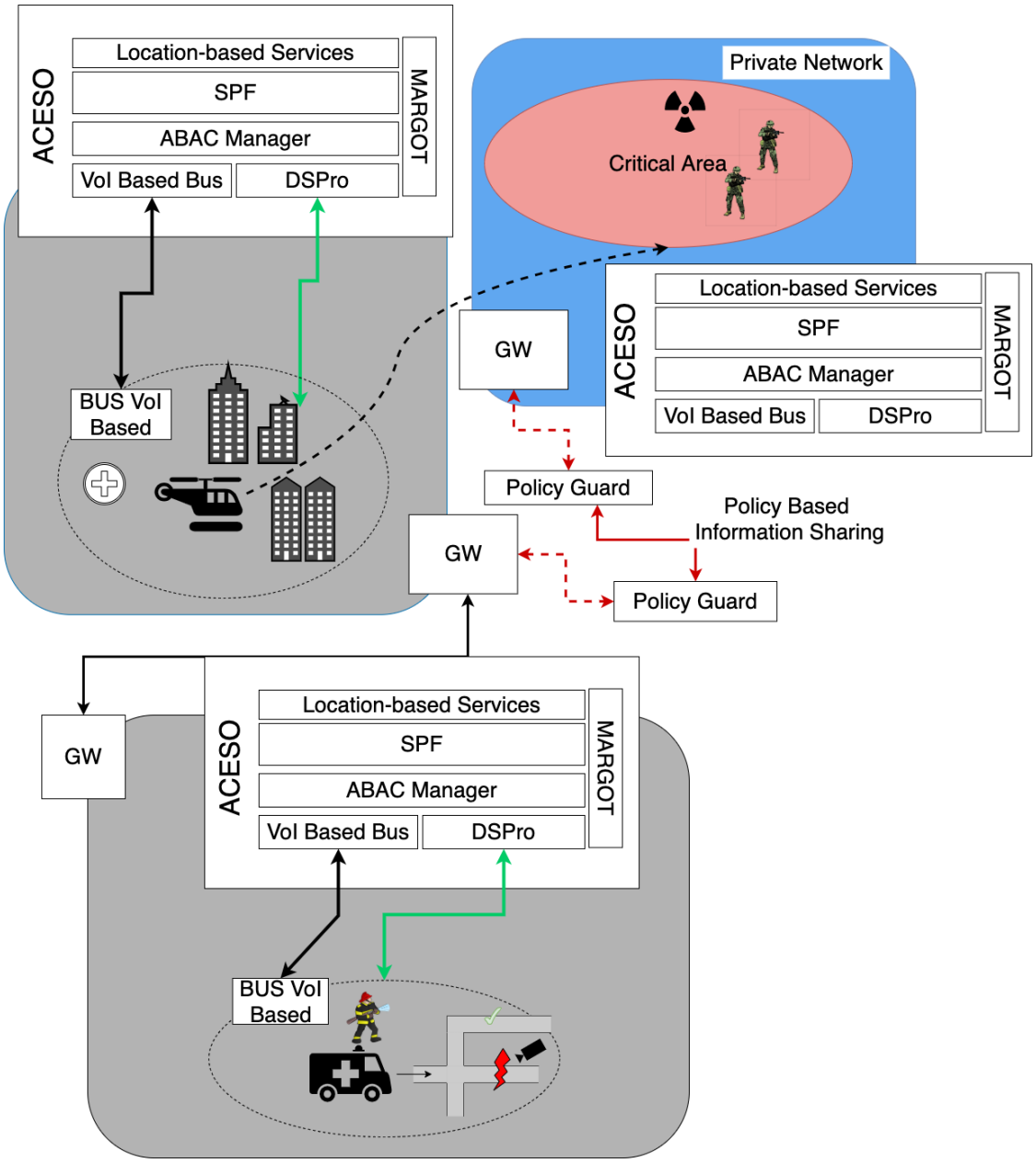


Figure 9.3: Multi-domain scenario with public and private domains

helicopter is responsible for monitoring the conditions of the city and for providing useful information to the other rescuers involved in the operations. Finally, the military team represents an adaptable force that can either bring its support and collaboration to the other rescuers or perform specific and entrusted tasks inside the blue domain represented in Fig. 9.3.

ACESO takes into account all these differences during the dispatching of information. Each time a user sends a request shares metadata related to its position (GPS coordinates), the information related to its role, authority affiliation, and other additional information required to evaluate the VoI of the specific request. For example, the team flying on the helicopter has a scarce interest in the road

traffic conditions. Instead, it is more interested in receiving information related to the air condition (environmental information), the presence of smoke clouds and the presence of collapsed buildings.

ACESO uses the metadata sent by applications to evaluate the VoI of each request, thus enabling it to tailor service responses based on the current user location, roles, and security policies as described in eq. (9.1). These attributes allow ACESO to dispatch only relevant information matching the users' interests, location, and affiliation. In this way, when firefighters are moving between areas, ACESO will assign higher VoI to all information about road traffic conditions, environmental data, and information related to areas that present favorable conditions for the possible spawn of new fires in the city. Meanwhile, the medical teams can also be provided with information about road traffic conditions. However, they will receive more data about their specific role, such as the presence of nearby hospitals, the number of victims that can be hosted or information regarding medical supplies required to stabilize the civilian victims.

Besides, Fig. 9.3 depicts how each domain is provided with a gateway (GW in the figure). Gateways allow information sharing between domains. To this end, the instances of ACESO running in the different domains exchange status messages that contain the roles of the users operating within the single domain. These roles allow ACESO to decide if the information generated in a domain may be useful to users operating in the other domains. In the case of inter-domain communications, the VoI evaluation adopts the PRD and TRD functions described in Section 9.1 to avoid geofencing information relevant to multiple domains.

Finally, the Policy Guard visible in Fig. 9.3 allows ACESO to evaluate the information labels related to the security policies for allowing or denying the release of information. This is a mandatory requirement for CIMIC in HADR operations and it is usually implemented using guards or data diodes to avoid information leakage from military systems to unauthorized users in the civilian domain. A guard can also restrict the information flow from civilian to military domains, thus providing some protection against malicious and untrusted information.

9.6 Simulation Results

To validate the effectiveness of ACESO in dealing with the HADR operations described in Section 9.5, an experimental evaluation over a fictional scenario within the Phileas simulator [148] is given. The fictional scenario is set in the city of Helsinki, Finland, which has been proven by NATO Group IST 147 to be the reference Smart City location for evaluation [149].

As described in Section 9.5, the scenario presents three different information

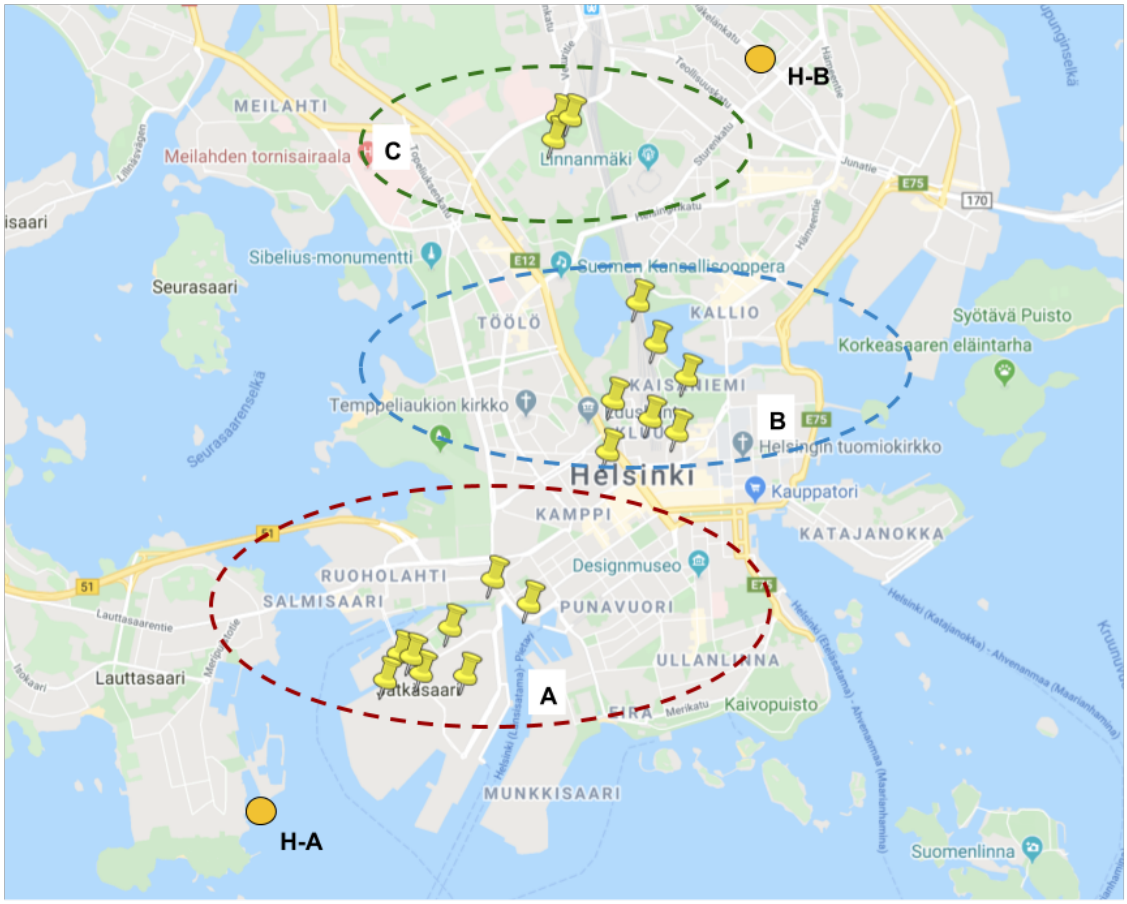


Figure 9.4: A map depicting the reference HADR scenario in the city of Helsinki, Finland.

domains defined as A, B, and C visible in Fig. 9.4. Along with these domains, the scenario defines four different types of users involved in the HADR operations with the following user types: a helicopter, a firefighter team, a medical team, and a military platoon.

Phileas is configured to reenact the described scenario for 24 hours for collecting the data from 18 data sources, and 6 edge devices dislocated within the three domains visible in Fig. 9.4. An important part during the simulation is the mobility of HADR operators along the different domains. In fact, it is essential to evaluate if the extended definition of VoI can provide inter-domain routing of context-aware information. To this end, the helicopter moves from location *H-A* to location *H-B* (visible in Fig. 9.4) following a line that crosses all domains. More involved in the pragmatical side of operations, the firefighter team covers both domains A and B. These firefighters will operate inside one or more vehicles that move following a straight route from domain A to domain B. Instead, the medical team still operates inside domains A and B, but their movements are simulated using a random walk mobility model with geographical boundaries covering both domains. Then, the mobility of the military platoon also follows a random walk mobility model within

Table 9.1: Roles labeling

Roles	Required services
Helicopter	BS, FD, EI
Firefighters	NS, FD, EI
Medical team	NS, MI, EI
Military team	CI, EI

domain C.

The locations of IoT sensors and the six edge devices in the city are visible as yellow pins in Fig. 9.4. The raw data collected from IoT sensors serve as an input for the services. At the beginning of the simulation, six services are running on the devices at the edge, analyzing data and disseminating the results in real-time: Navigation Service (NS), Fire Detection (FD), Building Status (BD), Environmental Info (EI), Medical Information (MI), and Classified Information (CI). NS collects video frames from traffic cameras and processes them, respectively, to assess the current viability status. FD also processes the video frames to identify/detect fires in the surrounding buildings, which are also monitored by the BD service that uses a combination of video frames and sensor data to look for fallen buildings. Instead, EI collects environmental info to detect the quality of air and other dangerous situations such as gas leaks. Finally, the MI service informs the medical operators of the injuries caused by the natural disaster to the citizens and their locations.

With regards to roles, the team flying on the helicopter is mainly interested in collecting information regarding fires and fallen buildings provided by the FD and BS services. Firefighters need to acquire information regarding the location of fires alongside their route and the information to reach those fires using the NS and FD services. The medical team is responsible for intervening to bring its support in the emergencies indicated by the medical information (MI) service. To do so, the medical team uses the navigation service (NS) to get a safe path to the position of the injured people. Finally, the military platoon is interested only in classified information (CI) that cannot be shared with the other teams for security reasons. As for the other teams, the military platoon collects environmental information leveraging on the EI service. To this end, Table 9.1 contains a detailed representation of the information labeling, which assigns to each team the services they require for the HADR operations.

On the one hand, domains A and B provide location-aware services for the civilian (firefighters and medical team) involved in the HADR operations. On the other hand, domain C is characterized by the presence of military platoon involved in the operations. While some civilian services running within domain C (EI, MI) provides useful information for the other teams, the information produced by military-specific

Table 9.2: Characterization of service processing.

Service Name	*TRD* half-life	*PRD* half-life
Environmental Info (EI)	2000 s	1 km
Fire Detection (FD)	5000 s	10 km
Building Status (BS)	15000 s	10 km
Navigation Service (NS)	5000 s	7.5 km
Medical Information (MI)	8000 s	10 km
Classified Information (CI)	1000 s	2.5 km

Table 9.3: Simulated bandwidth consumption for raw-data messages in the overall scenario.

Msg Type	Avg Msg Size)	Generated (MB)	Sent (%)
Medical	128 KB	6500 MB	79.34%
Video	756 KB	82300 MB	54.69%
Environmental	128 KB	3248 MB	54.59%
Pollution	1 KB	34 MB	64.72%
Temperature	256 B	17 MB	41.18%

services is not forwarded to the other domains for the security reasons stated above. The following simulation focuses on the capabilities of ACESO in dealing with information dispatching between the different roles and domains. With regards to the services, each one is defined with a linear TRD and PRD decay profile. More in detail, the TRD and PRD decay profiles of the location-aware services running in the scenario have been configured accordingly to Table 9.2.

Tab. 9.3 illustrates the benefits provided by the VoI Based Bus in filtering low value raw-data messages. In particular, Tab. 9.3 reports for each message type an average message size, the total data generated, and the percentage of transmitted data. It is worth noting how VoI filtering reduces the amount of transmitted data discarding the less valuable information to prioritize the processing and dissemination of the most valuable raw-data messages. The filtering is essential to reduce both bandwidth consumption and then to process delay at the fog nodes.

On the other hand, the aggregated VoI produced by the running services is illustrated in Fig. 9.5, which represents the VoI value of the aggregated IOs (generated during the simulation) through a box plot representation. More specifically, it is worth noting how some services have a higher VoI, thus indicating that the dispatch of this information will be privileged to whom services with a lower VoI.

Another interesting information regarding the implementation of location-aware services is the decay information is subjected to when travelling to domains. In fact, the VoI formulation defined in Section 9.4.1 allows to define for each service its domains of interest, thus allowing IOs to be forwarded and delivered to other domains when necessary. This is illustrated in Fig. 9.6, which depicts the VoI decay

profile for IOs messages (delivered to end-users) as a function of distance. On the whole, Fig. 9.6 depicts that each service has its own decay profile, i.e. the VoI is strictly dependent on the message class. It is worth noting that, *the definition of the PRD and TRD decay profiles enables a location-aware distribution of services.*

In addition, Fig. 9.6 demonstrates the effectiveness of the ACESO model in delivering information only to those users interested in consuming it that are within a defined domain (*geofencing*), e.g. classified information, as defined by the labeling depicted in Table 9.1. However, when information is relevant to other domains, ACESO can forward it to the to users, operating in other domains, whose roles match the information content. Services such as fire detection can have a minor decay in space and time, thus indicating that this type of information will likely be forwarded to the other operational domains.

Fig. 9.7 depicts that some information is delivered only to those users having a specific role and how the same piece of information can have different VoI values depending on the specific role. For example, IOs produced by the BS services are delivered only to the team flying on the helicopter. In contrast, IOs produced by the navigation service are delivered to both firefighters and the medical team but with different VoI. Finally, classified IOs are delivered only to the military platoon because other users cannot access the classified content.

Furthermore, Fig. 9.8 shows the amount of IO messages delivered every 10 minutes during the simulation time. It is worth to note that the medical information service delivers a constant and considerable amount of IOs since it is predictable that a large number of medical reports will be generated due to injuries caused by the consequence of the natural disaster. In addition, the 3 hours time-window with no messages in the BS histograms occurs when the helicopter moves to domain BS, where these IOs are not relevant.

Instead, Fig. 9.9 illustrates the importance of inter-domain routing in the overall

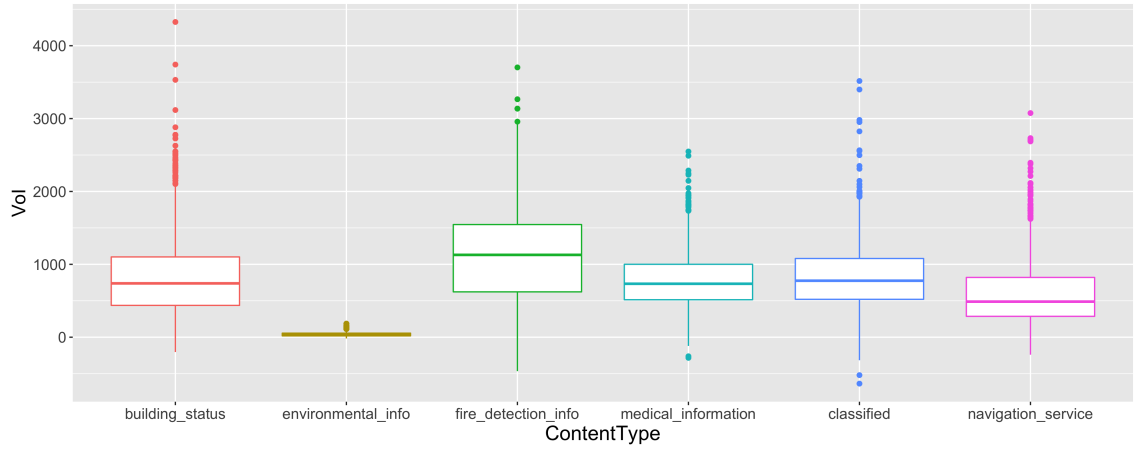


Figure 9.5: The VoI box plots of each content type during the simulation time.

Figure 9.6: VoI as a function of distance for IO messages.

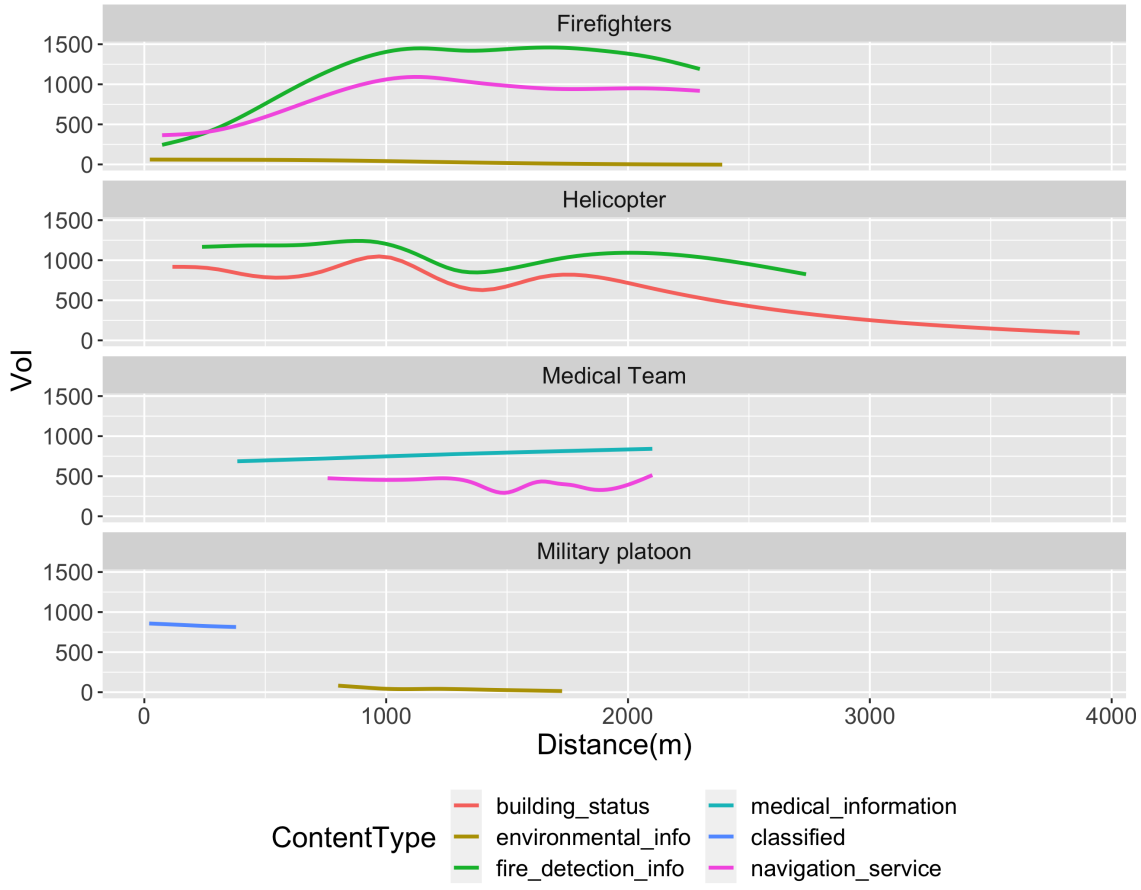


Figure 9.7: VoI trend as function of distance and role.

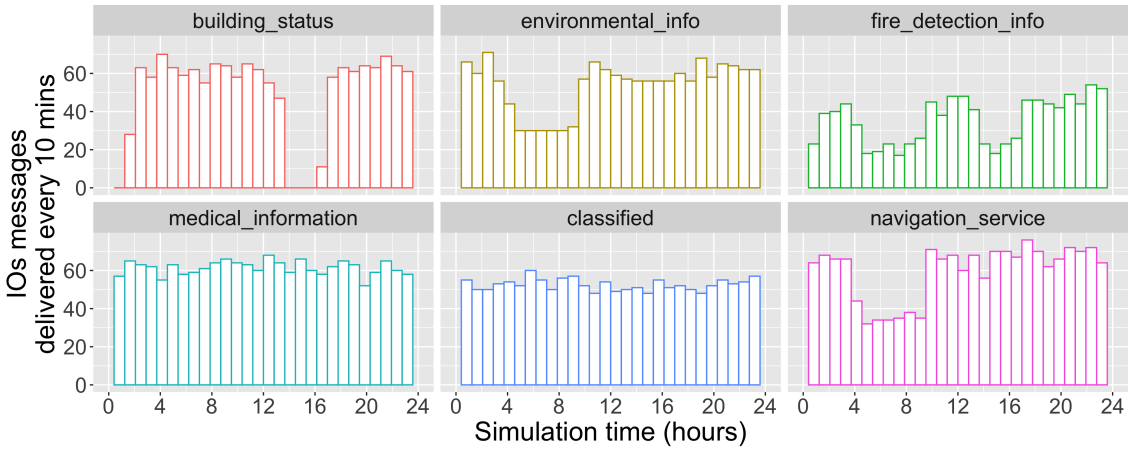


Figure 9.8: IOs delivered to end-users every 10 mins during simulation time.

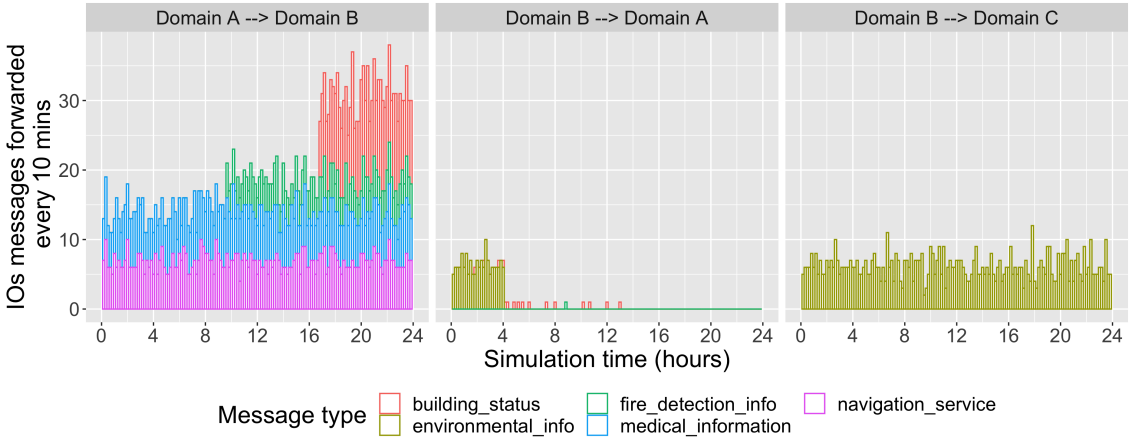


Figure 9.9: IOs forwarded between ACESO instances every 10 mins during simulation time.

ACESO architecture for creating Global Situational Awareness. This allows the firefighter team operating in domain A to be informed about viability information in domain B and so on.

Fig. 9.9 also shows how ACESO can act as data diode by prohibiting the forwarding of information matching defined security policies (classified information). In fact, the ACESO instance running in domain C does not release classified information to the other domains because of the security policies. Finally, it is worth pointing out that information is not simply replicated across all domains, but it is forwarded only when needed, e.g. the Fire Detection IOs are forwarded from Domain A to Domain B only when the firefighter team is within Domain B.

Chapter 10

Conclusion

Fog and Edge Computing are very interesting research topics in scientific literature, as they aim to enable a multitude of use-cases and to provide immersive, latency-sensitive, and real-time services. However, the development of Fog and Edge applications capable of satisfying such requirements calls for novel solutions capable to address the IoT data deluge and the management of scarce resources at the edge of the network.

To contribute to Fog and Edge Computing, this thesis investigated novel methodologies and tools to propose comprehensive middlewares for dealing with the challenges of Fog and Edge Computing environments. These middlewares provide valuable tools to address several use-case scenarios from Smart City applications to Humanitarian and Assistance Disaster Relief (HADR) operations.

To enable the management of Fog and Edge applications, this thesis presented the SPF middleware, which allows service providers to coordinate and distribute the service processing among a pool of devices located along the Cloud-IoT continuum. Then, with the purpose of enabling the simulation of these middlewares in realistic scenarios, this thesis presented the Phileas Simulator. Both SPF and Phileas make use of an innovative service model that defines Fog and Edge services as the composition of multiple service components: software modules that naturally integrates VoI techniques for filtering only the most valuable data in the processing and dissemination. Leveraging on this model, this thesis proposed different optimization techniques for maximizing the VoI utility delivered to the end-users of applications, ranging from simulation-based optimization to Deep Reinforcement Learning.

Then, this thesis presented middlewares that integrate the discussed methodologies and tools to achieve holistic management of applications and networks in Fog and Edge Computing. More specifically, these middlewares integrate VoI methodologies and tools by default to manage services and resources in a way that the VoI delivered to the end-users of applications is maximized. This is to achieve better use of the scarce resources available at the edge and enabling effective Fog and Edge

Computing.

To give a more exhaustive summary of the content of this thesis, Chapter 2 provided background on Fog and Edge Computing. Chapter 3 presented the concept of Value-of-Information (VoI) as novel criteria for Fog and Edge service modeling and resource management. To do so, this Chapter presented a novel notation for modeling the allocation of service components, composable service blocks, on devices.

Then, Chapter 4 presents two valuable tools built on the top of the AIV model, the SPF middleware and the Phileas simulator. On the one hand, SPF is a functional platform for running value-based services. On the other hand, Phileas is a discrete event simulator specifically designed for reenacting value-based Fog services. Both are valuable tools for the research presented in this thesis and they will be used for the implementation and the evaluation of the middlewares presented in the next Chapters.

Next, Chapter 5 discusses user-agnostic and user-specific VoI optimization models for maximizing the total VoI generated by Fog services. Furthermore, this Chapter presented novel contributions in formalizing user-specific VoI models that take into account the end-user specific utility in the VoI definition. Finally, Chapter 5 reported the adoption of optimization techniques for the VoI maximization of service components configurations on Fog devices.

Further investigating VoI optimization, Chapter 6 reported the research on Reinforcement Learning (RL) as another optimization tool for VoI based model. With this goal, Chapter 6 presented FogReinForce, a DRL based algorithm that trains a software agent to find the service components configuration maximizing the total VoI. Moreover, this Chapter discussed a possible implementation of a continuous optimization framework that relies on the FogReinForce algorithm to address the high dinamicity of Fog Computing.

After having formalized a resource management criterion and optimization tools for its maximization, this thesis proposed middlewares as valuable solutions for dealing with the complexity of Fog Computing management. Within this context, this Thesis presented the HORNET solution in Chapter 7 for enabling holistic management of Fog Computing. More specifically, HORNET leverages VoI both for services and network management by adopting a Multi-Layer Routing approach that considers multiple routing options to address different services' requirements.

Then, this thesis investigated the application of Fog Computing middlewares for dealing with Humanitarian and Disaster Relief (HADR) operations in Smart Cities. To this end, Chapter 8 presented the extended architecture of the SPF middleware that would enable easier integration with civilian IoT assets already presented in Smart Cities and military assets appositely deployed for the operations. The extended architecture is also to facilitate the management of applications and

services by HADR operators that need a software interface to monitor and interact in the ongoing operations.

Next, Chapter 9 presented ACESO, a proof of concept VoI middleware designed to support HADR operations in smart cities. ACESO implements a *breaking glass* policy that activates in emergency conditions, relaxing access control on resource utilization and enabling resilient communications. Furthermore, this Chapter presented and extended the VoI model incorporating the concepts of domains of interest and roles for supporting context- and location-aware services. The extended VoI model is evaluated within the Phileas simulator over a fictional HADR scenario involving multiple HADR teams with different roles.

On the whole, the application of VoI methodologies and tools into the design of Fog and Edge Computing middlewares delivered promising results. The researches illustrated in this thesis present novelties over the state of-the-art, as they address Fog and Edge Computing using the innovative criterion of VoI. Moreover, the presented efforts aim to give a comprehensive overview of the research on VoI from its formulation to its application in the design of value-based based Smart City middlewares. In addition, this thesis provides a step forward into the application of Fog and Edge Computing solutions to appealing scenarios such as HADR operations. Finally, the results achieved during this thesis demonstrated the soundness of the illustrated approaches, thus motivating further research into the adoption of these methodologies.

Future works will further investigate the topics presented during this thesis with the objective of improving the self-learning capabilities of the proposed middlewares. To this end, a future research direction is to compare different DRL algorithms in solving the VoI allocation problem. Finally, another research direction is to investigate service scalability at the device level for enabling elastic management of resources at the edge side of the network. Both are interesting research directions that complement the efforts presented in this thesis.

Concluding, this thesis summarized part of my research efforts achieved during PhD's program at the Department of Engineering of the University of Ferrara. During this program, I was part of the Distributed Systems Research Group under the supervision of my tutor Prof. Cesare Stefanelli and my co-tutor Prof. Mauro Tortonesi.

Bibliography

[1] M. Aazam, K. A. Harras, and S. Zeadally, "Fog computing for 5g tactile industrial internet of things: Qoe-aware resource allocation model," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3085–3092, May 2019.

[2] T. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017, cited By 307.

[3] 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.

[4] J. C. Nobre, A. M. de Souza, D. Rosário, C. Both, L. A. Villas, E. Cerqueira, T. Braun, and M. Gerla, "Vehicular software-defined networking and fog computing: Integration and design principles," *Ad Hoc Networks*, vol. 82, pp. 172 – 181, 2019.

[5] F. Bonomi, R. Milito, J. Zhu, and 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 2012*. ACM Press, 2012. [Online]. Available: https://doi.org/10.1145/2342509.2342513

[6] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, "The internet of things, fog and cloud continuum: Integration and challenges," *Internet of Things*, vol. 3-4, pp. 134 – 155, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2542660518300635

[7] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of*

Systems Architecture, vol. 98, pp. 289–330, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.sysarc.2019.02.009

[8] O. Salman, I. Elhajj, A. Chehab, and A. Kayssi, "Iot survey: An sdn and fog computing perspective," *Computer Networks*, vol. 143, pp. 221 – 246, 2018.

[9] L. Zhang and J. Li, "Enabling robust and privacy-preserving resource allocation in fog computing," *IEEE Access*, vol. 6, pp. 50 384–50 393, 2018.

[10] Y. Yin, W. Zhang, Y. Xu, H. Zhang, Z. Mai, and L. Yu, "Qos prediction for mobile edge service recommendation with auto-encoder," *IEEE Access*, vol. 7, pp. 62 312–62 324, 2019.

[11] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, Oct 2017.

[12] O. Consortium, "OpenFog Reference Architecture for Fog Computing," https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf, accessed: 11-15-2020.

[13] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[14] P. Bellavista, D. Belli, S. Chessa, and L. Foschini, "A social-driven edge computing architecture for mobile crowd sensing management," *IEEE Communications Magazine*, vol. 57, no. 4, pp. 68–73, April 2019.

[15] J. Santos, T. Wauters, B. Volckaert, F. De Turck, "Fog computing: Enabling the management and orchestration of smart city applications in 5g networks," *MDPI Entropy*, vol. 20, no. 1, 2018.

[16] R. A. Howard, "Information value theory," *IEEE Transactions on Systems Science and Cybernetics*, vol. 2, no. 1, pp. 22–26, Aug 1966.

[17] D. Turgut and L. Boloni, "Value of information and cost of privacy in the internet of things," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 62–66, Sep. 2017.

[18] L. Bölöni and D. Turgut, "Value of information based scheduling of cloud computing resources," *Future Generation Computer Systems*, vol. 71, pp. 212 – 220, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X16304472

[19] M. Giordani, T. Higuchi, A. Zanella, O. Altintas, and M. Zorzi, "A framework to assess value of information in future vehicular networks," in *Proceedings of the 1st ACM MobiHoc Workshop on Technologies, mOdels, and Protocols for Cooperative Connected Cars*, ser. TOP-Cars '19. New York, NY, USA: ACM, 2019, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/3331054.3331551

[20] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the internet of things," *Pervasive and Mobile Computing*, vol. 52, pp. 71–99, Jan. 2019. [Online]. Available: https://doi.org/10.1016/j.pmcj.2018.12.007

[21] 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.

[22] J. R. Michaelis, "Value of information driven content management in mixed reality infrastructures," vol. 10653, 2018. [Online]. Available: https://doi.org/10.1117/12.2303997

[23] ETSI, "Etsi and openfog consortium collaborate on fog and edge applications," *ETSI*. [Online]. Available: https://www.etsi.org/newsroom/news/1216-2017-09-news-etsi-and-openfog-consortium-collaborate-on-fog-and-edge-applications

[24] 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.

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

[26] Y. Bi, G. Han, C Lin, Q. Deng, L, Guo, F. Li, "Mobility support for fog computing: An sdn approach," *IEEE Communications Magazine*, vol. 56, pp. 53–59, May 2018.

[27] CISCO, "Perspectives iot, from cloud to fog computing," *CISCO*. [Online]. Available: https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing

[28] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 1826–1857, thirdquarter 2018.

[29] R. Buyya and S. N. Srirama, *Fog and edge computing: principles and paradigms*. John Wiley & Sons, 2019.

[30] M. Chiang, S. Ha, F. Risso, T. Zhang, and I. Chih-Lin, "Clarifying fog computing and networking: 10 questions and answers," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 18–20, 2017.

[31] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for internet of things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77 – 86, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864817301335

[32] I. S. Moskowitz, S. Russell, and N. Suri, "Chapter 9 - the value of information and the internet of thingsa," in *Artificial Intelligence for the Internet of Everything*, W. Lawless, R. Mittu, D. Sofge, I. S. Moskowitz, and S. Russell, Eds. Academic Press, 2019, pp. 145 – 169. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128176368000090

[33] M. Tortonesi, M. Govoni, A. Morelli, G. Riberto, C. Stefanelli, and N. Suri, "Taming the iot data deluge: An innovative information-centric service model for fog computing applications," *Future Generation Computer Systems*, vol. 93, pp. 888 – 902, 2019.

[34] Y. Mao, C. You, J. Zhang, K. Huang, and K. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017, cited By 879.

[35] Z. Guizani and N. Hamdi, "Cran, h-cran, and f-ran for 5g systems: Key capabilities and recent advances," *International Journal of Network Management*, vol. 27, no. 5, p. e1973, 2017, e1973 nem.1973. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.1973

[36] F. Poltronieri, C. Stefanelli, N. Suri, and M. Tortonesi, "Phileas: A simulation-based approach for the evaluation of value-based fog services," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Sep. 2018, pp. 1–6.

[37] 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.

[38] G. Benincasa, A. Morelli, C. Stefanelli, N. Suri, and M. Tortonesi, "Agile communication middleware for next-generation mobile heterogeneous networks," *IEEE Software*, vol. 31, no. 2, pp. 54–61, 2014.

[39] S. Margariti, V. Dimakopoulos, and G. Tsoumanis, "Modeling and simulation tools for fog computing-a comprehensive survey from a cost perspective," *Future Internet*, vol. 12, p. 89, 05 2020.

[40] 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.

[41] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 39–44.

[42] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[43] 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.

[44] I. Lera, C. Guerrero, and C. Juiz, "Yafs: A simulator for iot scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019.

[45] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey review*, vol. 23, no. 176, pp. 88–93, 1975.

[46] C. Thomas and W. Featherstone, "Validation of vincenty's formulas for the geodesic using a new fourth-order extension of kivioja's formula," *Journal of Surveying engineering*, vol. 131, no. 1, pp. 20–26, 2005.

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483–502, 2002.

[53] F. Poltronieri, M. Tortonesi, A. Morelli, C. Stefanelli, and N. Suri, "Value of Information based Optimal Service Fabric Management for Fog Computing," in *Proceedings of 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS 2020)*, Apr. 2020, pp. 1–9.

[54] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 339–367, Sep. 2016.

[55] T. Dasu, S. Krishnan, and G. M. Pomann, "Robustness of change detection algorithms," in *Advances in Intelligent Data Analysis X*. Springer Berlin Heidelberg, 2011, pp. 125–137.

[56] R. Killick, *Introduction to optimal changepoint detection algorithms*, 2017 (accessed June 21st, 2019). [Online]. Available: http://members.cbio.mines-paristech.fr/~thocking/change-tutorial/RK-CptWorkshop.html

[57] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes - Theory and Application*, 1993 (accessed June 21st, 2019). [Online]. Available: http://people.irisa.fr/Michele.Basseville/kniga/

[58] A. Gosavi, *Simulation-Based Optimization*, 2nd ed. Springer US, 2015.

[59] "NS3, Logarithmic Distance Propagation Loss Model," https://www.nsnam.org/doxygen/classns3_1_1_log_distance_propagation_loss_model.html, accessed: 03-07-2020.

[60] C. Canali and R. Lancellotti, "A fog computing service placement for smart cities based on genetic algorithms," in *CLOSER*, 2019.

[61] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck, "Resource provisioning in fog computing: From theory to practice †," *Sensors*, vol. 19, no. 10, p. 2238, May 2019. [Online]. Available: https://doi.org/10.3390/s19102238

[62] F. Poltronieri, C. Stefanelli, N. Suri, and M. Tortonesi, "Phileas: A simulation-based approach for the evaluation of value-based fog services," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Sep. 2018, pp. 1–6.

[63] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[64] J. Sun, C.-H. Lai, and X.-J. Wu, *Particle swarm optimisation: classical and quantum perspectives*. Crc Press, 2016.

[65] K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran, M. N. Al-Mhiqani, A. A. Mutlag, S. A. Mostafa, N. S. Ali, and D. A. Ibrahim, "A review of fog computing and machine learning: Concepts, applications, challenges, and open issues," *IEEE Access*, vol. 7, pp. 153 123–153 140, 2019.

[66] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 712–725, 2019.

[67] F. Poltronieri, M. Tortonesi, C. Stefanelli, and N. Suri, "Reinforcement Learning for value-based Placement of Fog Services," in *(Submitted) IM 2021 IFIP/IEEE International Symposium on Integrated Network Management, Budapest, 17-21 May 2021*, 2021.

[68] I. Comșa, R. Trestian, G. Muntean, and G. Ghinea, "5mart: A 5g smart scheduling framework for optimizing qos through reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1110–1124, June 2020.

[69] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Bradford Books, 2018.

[70] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1318–1331, Dec 2019.

[71] H. Yao, S. Ma, J. Wang, P. Zhang, C. Jiang, and S. Guo, "A continuous-decision virtual network embedding scheme relying on reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 864–875, June 2020.

[72] A. Kaur and K. Kumar, "Energy-efficient resource allocation in cognitive radio networks under cooperative multi-agent model-free reinforcement learning schemes," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1337–1348, Sep. 2020.

[73] L. Mai, N.-N. Dao, and M. Park, "Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing," *Sensors*, vol. 18, p. 2830, 08 2018.

[74] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019, cited By 67.

[75] F. Wei, G. Feng, Y. Sun, Y. Wang, and Y.-C. Liang, "Dynamic network slice reconfiguration by exploiting deep reinforcement learning," vol. 2020-June, 2020, cited By 0.

[76] A. Gosavi, "Reinforcement learning: A tutorial survey and recent advances," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 2009.

[77] B. Dab, N. Aitsaadi, and R. Langar, "Q-learning algorithm for joint computation offloading and resource allocation in edge cloud," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 45–52.

[78] M. Nakanoya, Y. Sato, and H. Shimonishi, "Environment-adaptive sizing and placement of nfv service chains with accelerated reinforcement learning," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, April 2019, pp. 36–44.

[79] J. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2019, pp. 1–7.

[80] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018, cited By 37.

[81] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, pp. 10 – 17, 2019, artificial Intelligence for Future Wireless Communications and Networking. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864818301469

[82] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," *Aerospace Science and Technology*, vol. 92, pp. 588 – 594, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963818318704

[83] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[84] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[85] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[86] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[87] P. Bellavista, C. Giannelli, D. D. P. Montenero, F. Poltronieri, C. Stefanelli, and M. Tortonesi, "Holistic processing and networking (hornet): An integrated solution for iot-based fog computing services," *IEEE Access*, vol. 8, pp. 66 707–66 721, 2020.

[88] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, November, 2013.

[89] A. C. Baktir, A. Ozgovde, C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 4th quarter, 2017.

[90] R.Vilalta, et al., "Software defined network service chaining for ott service providers in 5g networks," *IEEE Communications Magazine*, vol. 55, pp. 124–131, November 2017.

[91] ——, "Telcofog: A unified flexible fog and cloud computing architecture for 5g networks," *IEEE Communications Magazine*, vol. 55, pp. 36–43, August 2017.

[92] P. K. Sharma, M. Chen, and J. H. Park, "A software defined fog node based distributed blockchain cloud architecture for iot," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[93] F. Y. Okay and S. Ozdemir, "Routing in fog-enabled iot platforms: A survey and an sdn-based solution," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4871–4889, Dec 2018.

[94] C. Li, Z. Qin, E. Novak, and Q. Li, "Securing sdn infrastructure of iot–fog networks from mitm attacks," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1156–1164, Oct 2017.

[95] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, "Fog-assisted sdn controlled framework for enduring anomaly detection in an iot network," *IEEE Access*, vol. 6, pp. 73 713–73 723, 2018.

[96] W. Villota, M. Gironza, A. Ordoñez, and O. M. Caicedo Rendon, "On the feasibility of using hierarchical task networks and network functions virtualization for managing software-defined networks," *IEEE Access*, vol. 6, pp. 38 026–38 040, 2018.

[97] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[98] R. Yu, G. Xue, and X. Zhang, "Qos-aware and reliable traffic steering for service function chaining in mobile networks," *IEEE Journal on Selected Areas in Comm.*, vol. 35, no. 11, pp. 2522–2531, 2017, cited By 3.

[99] X. Zhang, S. Yu, J. Zhang, and Z. Xu, "Forwarding rule multiplexing for scalable sdn-based internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3373–3385, April 2019.

[100] H. Hawilo, M. Jammal, and A. Shami, "Network function virtualization-aware orchestrator for service function chaining placement in the cloud," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 643–655, 2019, cited By 1.

[101] D. Zhao, J. Ren, R. Lin, S. Xu, and V. Chang, "On orchestrating service function chains in 5g mobile network," *IEEE Access*, vol. 7, pp. 39 402–39 416, 2019.

[102] L. M. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal-Bernabe, S. N. Srirama, and M. F. Zhani, "Research challenges in nextgen service orchestration," *Future Generation Computer Systems*, vol. 90, pp. 20 – 38, 2019.

[103] H. Gao, W. Huang, Y. Duan, X. Yang, and Q. Zou, "Research on cost-driven services composition in an uncertain environment," *Journal of Internet Technology*, vol. 20, no. 3, pp. 755–769, 2019.

[104] H. Gao, Y. Duan, L. Shao, and X. Sun, "Transformation-based processing of typed resources for multimedia sources in the iot environment," *Wireless Networks*, Nov 2019.

[105] Y. Chen, S. Deng, H. Ma, and J. Yin, "Deploying data-intensive applications with multiple services components on edge," *Mobile Networks and Applications*, Apr. 2019.

[106] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for iot devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73 520–73 530, 2018.

[107] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware qos prediction with neural collaborative filtering for internet-of-things services," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[108] F. Poltronieri, M. Tortonesi, A. Morelli, C. Stefanelli, and N. Suri, "Value of information based optimal service fabric management for fog computing," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[109] P. Bellavista, A. Corradi, and C. Giannelli, "Middleware for differentiated quality in spontaneous networks," *IEEE Pervasive Computing*, vol. 11, no. 3, pp. 64–75, March 2012.

[110] C. Giannelli, P. Bellavista, D. Scotece, "Software defined networking for quality-aware management of multi-hop spontaneous networks," in *Proceedings of Int. Conf. on Computing, Networking and Communications (ICNC 2018), Maui, Hawaii, USA, March 5-8, 2018*. IEEE, 2018.

[111] P. Bellavista, A. Dolci, C. Giannelli, "Manet-oriented sdn: Motivations, challenges, and a solution prototype," in *19TH IEEE Int. Symp. on a World*

of Wireless, Mobile and Multimedia Networks (WoWMoM 2018), Chania, Greece, June 12-15, 2018. IEEE, 2018.

[112] J. Yu, M. Tan, H. Zhang, D. Tao, and Y. Rui, "Hierarchical deep click feature prediction for fine-grained image recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[113] J. Yu, J. Li, Z. Yu, and Q. Huang, "Multimodal transformer with multi-view visual representation for image captioning," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2019.

[114] S. M. Libelli and P. Alba, "Adaptive mutation in genetic algorithms," *Soft Computing*, vol. 4, no. 2, pp. 76–80, Jul. 2000.

[115] H. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," Naval Research Lab, Washington DC, USA, Tech. Rep. AIC-90-001, 1990.

[116] G. Riberto, M. Govoni, C. Stefanelli, N. Suri, and M. Tortonesi, "Leveraging civilian iot infrastructures to support warfighting activities in urban environments," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Feb 2018, pp. 118–123.

[117] M. Pradhan, F. Poltronieri, and M. Tortonesi, "Dynamic resource discovery and management for edge computing based on spf for hadr operations," in *2019 International Conference on Military Communications and Information Systems (ICMCIS)*. IEEE, 2019, pp. 1–6.

[118] ——, "Generic architecture for edge computing based on spf for military hadr operations," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 225–230.

[119] T.-h. Kim, C. Ramos, and S. Mohammed, "Smart city and iot," *Future Generation Computer Systems*, vol. 76, pp. 159–162, 11 2017.

[120] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[121] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10 – 28, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517301455

[122] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[123] H. Ahvenniemi, A. Huovila, I. Pinto-Seppä, and M. Airaksinen, "What are the differences between sustainable and smart cities?" *Cities*, vol. 60, pp. 234–245, Feb. 2017.

[124] H. Baer and M. Singer, *Global warming and the political ecology of health: Emerging crises and systemic solutions*. Routledge, 2016.

[125] F. T. Johnsen, Z. Zieliński, K. Wrona, N. Suri, C. Fuchs, M. Pradhan, J. Furtak, B. Vasilache, V. Pellegrini, M. Dyk, M. Marks, and M. Krzysztoń, "Application of iot in military operations in a smart city," in *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, 2018, pp. 1–8.

[126] M. Pradhan, N. Suri, C. Fuchs, T. H. Bloebaum, and M. Marks, "Toward an architecture and data model to enable interoperability between federated mission networks and iot-enabled smart city environments," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 163–169, 2018.

[127] B. Taylor, V. S. Marco, W. Wolff, Y. Elkhatib, and Z. Wang, "Adaptive deep learning model selection on embedded systems," *SIGPLAN Not.*, vol. 53, no. 6, p. 31–43, Jun. 2018. [Online]. Available: https://doi.org/10.1145/3299710.3211336

[128] J. Flotyński, K. Krysztofiak, and D. Wilusz, "Building modular middlewares for the internet of things with osgi," in *The Future Internet*, A. Galis and A. Gavras, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 200–213.

[129] C. Vallati, E. Mingozzi, G. Tanganelli, N. Buonaccorsi, N. Valdambrini, N. Zonidis, B. Martínez, A. Mamelli, D. Sommacampagna, B. Anggorojati, S. Kyriazakos, N. Prasad, F. J. Nieto, and O. B. Rodriguez, "Betaas: A platform for development and execution of machine-to-machine applications in the internet of things," *Wirel. Pers. Commun.*, vol. 87, no. 3, p. 1071–1091, Apr. 2016. [Online]. Available: https://doi.org/10.1007/s11277-015-2639-0

[130] K. Wrona, M. Tortonesi, M. Marks, and N. Suri, "Leveraging and fusing civil and military sensors to support disaster relief operations in smart environments," in *2019 AFCEA/IEEE Military Communications Conference (MIL-*

COM 2019) - Special Session on NATO Interoperability and Information Sharing. IEEE, Nov. 2019, pp. 1–6.

[131] NATO Civil Emergency Planning Euro-Atlantic Disaster Response Coordination Centre, "NATO's Role in Disaster Assistance," NATO, Tech. Rep., 05 2000. [Online]. Available: https://www.nato.int/eadrcc/mcda-e.pdf

[132] T. hoon Kim, C. Ramos, and S. Mohammed, "Smart city and iot," *Future Generation Computer Systems*, vol. 76, pp. 159 – 162, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17305253

[133] V. Vescoukis, N. Doulamis, and S. Karagiorgou, "A service oriented architecture for decision support systems in environmental crisis management," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 593–604, 2012, cited By 64.

[134] J. Sánchez-García, D. Reina, and S. Toral, "A distributed pso-based exploration algorithm for a uav network assisting a disaster scenario," *Future Generation Computer Systems*, vol. 90, pp. 129–148, 2019, cited By 27.

[135] S. Sharafeddine and R. Islambouli, "On-demand deployment of multiple aerial base stations for traffic offloading and network recovery," *Computer Networks*, vol. 156, pp. 52–61, Jun. 2019. [Online]. Available: https://doi.org/10.1016/j.comnet.2019.03.016

[136] G. Deepak, A. Ladas, and C. Politis, "Robust device-to-device 5g cellular communication in the post-disaster scenario," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, Jan. 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ccnc.2019.8651832

[137] E. Sacoto-Cabrera, L. Guijarro, J. Vidal, and V. Pla, "Economic feasibility of virtual operators in 5g via network slicing," *Future Generation Computer Systems*, vol. 109, pp. 172–187, 2020, cited By 0. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85082671939&doi=10.1016%2fj.future.2020.03.044&partnerID=40&md5=63f7dd8cc376baa14dacd22152681aaf

[138] D. Satria, D. Park, and M. Jo, "Recovery for overloaded mobile edge computing," *Future Generation Computer Systems*, vol. 70, pp. 138–147, 2017, cited By 50.

[139] C. Cicconetti, M. Conti, A. Passarella, and D. Sabella, "Toward distributed computing environments with serverless solutions in mec systems," *IEEE Communications Magazine*, vol. 58, no. 3, 2020.

[140] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the internet of things," *Pervasive and Mobile Computing*, vol. 52, pp. 71 – 99, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119218301111

[141] M. Aazam, S. Zeadally, and K. Harras, "Offloading in fog computing for iot: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018, cited By 55.

[142] A. M. Nagy and V. Simon, "Survey on traffic prediction in smart cities," *Pervasive and Mobile Computing*, vol. 50, pp. 148 – 163, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119217306521

[143] D. J. Cook and S. K. Das, "Pervasive computing at scale: Transforming the state of the art," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 22 – 35, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119211001416

[144] L. Campioni, R. Lenzi, F. Poltronieri, M. Pradhan, C. Stefanelli, N. Suri, and M. Tortonesi, "MARGOT: Dynamic IoT Resource Discovery for HADR Environments," in *Proceedings of the 2019 AFCEA/IEEE Military Communications Conference (MILCOM 2019)*, Nov. 2019, pp. 1–6.

[145] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, Feb 2015.

[146] F. Poltronieri, L. Campioni, R. Lenzi, A. Morelli, N. Suri, and M. Tortonesi, "Secure multi-domain information sharing in tactical networks," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 1–6.

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

[148] F. Poltronieri, C. Stefanelli, N. Suri, and M. Tortonesi, "Phileas: A simulation-based approach for the evaluation of value-based fog services," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Sep. 2018, pp. 1–6.

[149] F. T. Johnsen, Z. Zieliński, K. Wrona, N. Suri, C. Fuchs, M. Pradhan, J. Furtak, B. Vasilache, V. Pellegrini, M. Dyk, M. Marks, and M. Krzysztoń, "Application of iot in military operations in a smart city," in *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, 2018, pp. 1–8.

Author's Publication List

Stefano Alvisi, Francesco Casellato, Marco Franchini, Marco Govoni, Chiara Luciani, Filippo Poltronieri, Giulio Riberto, Cesare Stefanelli, and Mauro Tortonesi. Wireless middleware solutions for smart water metering. *Sensors*, 19(8):1853, 2019.

Paolo Bellavista, Carlo Giannelli, Dmitrij David Padalino Montenero, Filippo Poltronieri, Cesare Stefanelli, and Mauro Tortonesi. Holistic processing and networking (hornet): An integrated solution for iot-based fog computing services. *IEEE Access*, 8:66707–66721, 2020.

Lorenzo Campioni, Rita Lenzi, Filippo Poltronieri, Manas Pradhan, Mauro Tortonesi, Cesare Stefanelli, and Niranjan Suri. Margot: Dynamic iot resource discovery for hadr environments. In *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*, pages 809–814. IEEE, 2019.

Carlo Giannelli, Filippo Poltronieri, Cesare Stefanelli, and Mauro Tortonesi. Supporting the development of next-generation fog services. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6. IEEE, 2018.

Pradhan Manas, Filippo Poltronieri, Mauro Tortonesi, et al. Generic architecture for edge computing based on spf for military hadr operations. In *IEEE 5th World Forum on Internet of Things (WF-IoT)-Special Session on Military Applications of IoT*, pages 1–6. IEEE, 2019.

Filippo Poltronieri, Lorenzo Campioni, Rita Lenzi, Alessandro Morelli, Niranjan Suri, and Mauro Tortonesi. Secure multi-domain information sharing in tactical networks. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 1–6. IEEE, 2018.

Filippo Poltronieri, Lorenzo Campioni, Lenzi Rita, Mantovani Mattia, Alessandro Morelli, Cesare Stefanelli, Suri Niranjan, Mauro Tortonesi, et al. Efficient and secure multi-domain information sharing in tactical networks. In *23rd International Command and Control Research and Technology Symposium (ICCRTS 2018)*, pages 1–10. International Command and Control Institute, 2018.

Filippo Poltronieri, Roberto Fronteddu, Cesare Stefanelli, Niranjan Suri, Mauro Tortonesi, Matthew Paulini, and James Milligan. A secure group communication approach for tactical network environments. In *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–8. IEEE, 2018.

Filippo Poltronieri, Laurel Sadler, Giacomo Benincasa, Timothy Gregory, John M Harrell, Somiya Metu, and Christine Moulton. Enabling efficient and interoperable control of iobt devices in a multi-force environment. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 757–762. IEEE, 2018.

Filippo Poltronieri, Cesare Stefanelli, Suri Niranjan, Mauro Tortonesi, et al. Analyzing and evaluating information-centric and value-based fog service architectures in military environments: The phileas simulator. In *23rd International Command and Control Research and Technology Symposium (ICCRTS 2018)*, pages 1–12. International Command and Control Institute, 2018.

Filippo Poltronieri, Cesare Stefanelli, Niranjan Suri, and Mauro Tortonesi. Phileas: A simulation-based approach for the evaluation of value-based fog services. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6. IEEE, 2018.

Filippo Poltronieri, Mauro Tortonesi, Alessandro Morelli, Cesare Stefanelli, and Niranjan Suri. Value of information based optimal service fabric management for fog computing. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2020.

Niranjan Suri, Roberto Fronteddu, Eelco Cramer, Maggie Breedy, Kelvin Marcus, Ronald in't Velt, Jan Nilsson, Mattia Mantovani, Lorenzo Campioni, Filippo Poltronieri, et al. Experimental evaluation of group communications protocols for tactical data dissemination. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 133–139. IEEE, 2018.