Service Orchestration in Fog Environments

Karima Velasquez*, David Perez Abreu*, Diogo Gonçalves†,
Luiz Bittencourt†, Marilia Curado*, Edmundo Monteiro*, and Edmundo Madeira†
*CISUC, Department of Informatics Engineering
University of Coimbra, Portugal
Email: {kcastro,dabreu,marilia,edmundo}@dei.uc.pt
†Institute of Computing
University of Campinas, Brazil
Email: diogomg@lrc.ic.unicamp.br, {bit,edmundo}@ic.unicamp.br

***Abstract*—A new era of automated services has permeated user's daily lives thanks to paradigms such as Smart City and the Internet of Things. This shift from traditional applications is possible due to the massive amount of heterogeneous devices that constitute the Internet of Things. To provide newly improved characteristics to these services, such as mobility support, high resilience, and low latency, an extension to the Cloud computing paradigm was created, called Fog computing, which brings processing and storage resources towards the edge of the network, in the vicinity of the Internet of Things environment.**

This scenario implies a higher complexity level needed to coordinate available resources and how applications and services use them. Although some solutions have been proposed for the Cloud, several characteristics differentiate the Cloud from the Fog, creating the need for new mechanisms for the coordination of resources, applications, and services in the Fog.

This paper explains the challenges present in the Fog that call for new mechanisms to later propose an architecture to manage resources in the Fog using a hybrid approach. In the Internet of Things and South-Bound Fog Levels, a distributed management of applications and services is proposed applying choreography techniques to enable automated fast decision making. A centralized approach to orchestrate applications and services taking advantage of a global knowledge of the resources available in the network is suggested for the North-Bound Fog and Cloud Levels.

I. Introduction

The development of resource demanding applications, and the proliferation of applications within the context of the Smart City and Internet of Things (IoT) paradigms lead to the search of resources that, until certain point, was covered by the use of the Cloud computing paradigm. The Cloud represents a logically centralized pool of resources that are exploited by resource-hungry applications. However, this solution does not satisfy the needs of every application.

The Fog is an extension of the Cloud that brings resources closer to the users, to the edge of the network. This paradigm was conceived to address applications and services that do not fit well in the Cloud paradigm. Applications that require characteristics such as low latency, geo-distribution, mobility, high resilience, and large-scale distributed systems benefit from the Fog.

The Fog is located in the vicinity of the user, on the frontier with the IoT, where there is a plethora of heterogeneous devices that have to work harmoniously to keep the services running at acceptable Quality of Service (QoS) levels. This fact calls for the automation of management functions to deal with the complexity of the scenario. Hence, orchestration functions are required in order to automate the management in such a dense, heterogeneous, and complex environment.

Orchestration denotes a single centralized executable process that coordinates the interaction between different applications or services. Hence, orchestration uses a centralized approach to applications and services composition. Even though different orchestration approaches have been applied to other scenarios such as the Cloud, the Fog has particular characteristics, such as its distributed nature, the heterogeneity of its devices, and the constraints in their resources.

This calls for a new revision on orchestration solutions to adapt them to the Fog taking into consideration its characteristics. Furthermore, a review of decentralized choreography principles to add to the solutions, in order to provide dynamism and real-time response at the lower levels of the infrastructure, is also required. Choreography could be defined as a global description of the participating applications and services, which is specified by the exchange of information, rules of cooperation and agreements among two or more endpoints; thereby, this strategy uses a decentralized approach for applications and services composition.

The *Supporting the Orchestration of Resilient and Trustworthy Fog Services* (SORTS) project is aimed at the design, implementation, and development of a service orchestrator for Fog environments. The service orchestrator must be capable of maintaining resilience, trustworthiness, and low latency in a dynamic environment, such as the Fog, and also guaranteeing acceptable levels of QoS. The choreography support for the lower levels of the architecture will provide more dynamism to the proposed solution.

This paper presents an architecture for Fog environments management, taking into consideration a hybrid approach including both orchestration and choreography. Orchestration will be used on the North-Bound Region of the Fog and in the Cloud, in order to have a global view of the system. On the other hand, choreography will be used in the South-Bound Region of the Fog, enabling automated fast responses at the lower levels.

The contributions of this paper are the following:

1) The proposal of a hybrid approach for orchestration and choreography of services in the Fog;
2) An architecture for the orchestrator, following the proposed hybrid approach.

The paper is structured as follows. Section II presents a subset of related work in the orchestration and choreography fields. Section III describes the Fog scenario and some particular challenges that differentiate it from the Cloud. Section IV depicts the designed architecture for a Fog orchestrator, following a hybrid approach. Section V offers a description of a use-case where this orchestrator is framed, the SORTS project, including an example where the orchestrator could operate. Finally, Section VI concludes the paper.

II. Related Work

The concept of orchestration frequently comes around when discussing service oriented architectures, virtualization, and management of resources in network infrastructures. The need for automated procedures to deal with different topics has been explored before in diverse contexts. Zaalouk et al. [1] propose an orchestrator-based architecture that deals with security issues using some Software Defined Networking (SDN) features. The orchestrator is the core of the architecture proposed by the authors and is in charge of turning on and off the applications for detecting attacks on the system. Jaeger [2] describes an architecture using Network Function Virtualization (NFV) orchestration and management entities for a hybrid network. Jaeger's work is focused on extending the European Telecommunications Standards Institute (ETSI) NFV reference architecture to manage and orchestrate security functions.

The migration of Virtual Machines (VM) between different network domains using an SDN network orchestrator is addressed by Mayoral et al. [3]. The migration process involves several steps, including the requests from the VM, the handling of the active connections, and a final acknowledgment. To handle the amount of optical resources assigned to connections, Velasco et al. [4] present the architecture of an orchestrator that allows elastic data center operation as well as dynamic establishment and teardown of inter-datacenters connections.

Vilalta et al. [5] show a hierarchical SDN orchestration architecture for heterogeneous wireless and optical networks. The paper also introduces end-to-end provisioning and recovery procedures in a multi-domain network. Qin et al. [6] propose an architecture that allows a differentiation of quality levels for IoT tasks in different wireless scenarios. This architecture includes an SDN controller that differentiates flow scheduling over multi-hop heterogeneous ad-hoc paths. Martini et al. [7] describe an SDN-based orchestrator for computing and communication resources chaining; authors also provide an evaluation of different chaining strategies to compose a pool of resources based on variable load. Wen et al. [8] introduce a framework to orchestrate Fog computing environments. This framework manages the composition of IoT applications, using a genetic-based solution for the planning.

The works analyzed so far are too focused on dealing with single issues, such as security, connectivity, and planning. Additionally, these works do not consider the use of choreography together with orchestration to provide a dynamic solution, more compatible with real-time applications.

When using choreography, applications and services have direct communication among each other without any central entity coordinating them. Cherrier et al. [9] study the impact of using orchestration and choreography in Wireless Sensor and Actuator Networks (WSAN) using mathematical analysis and also application experiments. Their study shows the benefit of using choreography because of the shorter paths used for communication between the nodes and also concerning network reliability.

Pedraza and Estublier [10] propose to depict applications as a classic service orchestration extended by annotations describing where their activities are to be executed. Their proposal transforms the orchestration into a number of sub-orchestrations to be deployed on a set of choreography servers, to deploy and execute later on the application. This approach shows how to transform an orchestration into a distributed choreographed solution. Issues of description, orchestration, and choreography of web services using the Reo coordination language and constraint automata are addressed in the work of Meng and Arbab [11]. Based on the constraint automata, authors use a unifying abstract level allowing them to determine the relationship between orchestration and choreography.

Furtado et al. [12] introduce a middleware for choreography able to automatically deploy and execute services. The middleware is also responsible for monitoring the service composition execution and for performing automatic resource provisioning and service reconfiguration to achieve agreed QoS levels. The middleware allows for an automated service composition on which the responsibilities for execution are shared among the service components without a centralized point of coordination.

The reviewed works are focused in using orchestration or choreography (or how to go from one to the other), but not a hybrid approach. Moreover, most of them (except Wen et al. [8]) do not even consider the particular characteristics of Fog environments. Some challenges that ought to be considered by a Fog orchestrator are depicted in the section below.

III. Fog Environments and its Challenges

In recent years there has been a lot of discussion about Fog networks leading to some confusion regarding the actual meaning of the term [13]. Initially brought up by Cisco to describe the paradigm shift from the Cloud towards the edge of the network [14], nowadays there seems to be a widely adopted consensus on the concept. The Fog network extends the Cloud computing paradigm to the edge of the network, allowing the delivery of a new set of services and

applications that are not entirely fit for the Cloud [15][16], and providing them particular characteristics including ubiquity, decentralized management, high resilience, low latency, and mobility [17][18].

In the edge of the network, devices not only request services and data from the upper layer (Cloud) but also take care of some computing tasks (e.g. caching, processing) [19]. These devices are arranged into groups called *Cloudlets* [20][21], that constitute the Fog instances. A Cloudlet is a set of devices located in the vicinity of the mobile user that are used to offload applications when user devices are not capable of executing them [22]. The use of Cloudlets and Fog networks, in general, has proven to be beneficial in terms of improving response times and mobile energy consumption, among other factors [23].

As in the Cloud, there is a need to manage the resources in the Fog properly. Many solutions have already been developed with this purpose for the Cloud; but even though the Cloud and the Fog use the same kind of resources (storage, network, and processing) and share the need for many of the same mechanisms (e.g., virtualization), the same procedures can not be migrated from one environment to the other, given that there are fundamental differences among them [24][25].

Unlike the Cloud, which is centralized, the Fog aims at services and applications with *distributed* deployment [26]. The management of the devices in the micro datacenters between the Cloud and the underlying IoT is another requirement for the Fog [27][28]. Fog devices are placed between smart devices and the Cloud; in this area, the homogeneity of resources that can be found in the Cloud disappears to give room for a more *heterogeneous* environment, where the characteristics and manufacturers of the resources vary [29].

Furthermore, the devices in the Fog are *resource constrained* in comparison with the resource-rich Cloud counterpart [30]; energy, processing, and storage resources are limited in the Fog. The Fog also deals with *mobility*, where mobile and wireless nodes are commonly less reliable in their connectivity behavior [31]; this can cause drops in the Quality of Experience (QoE) or even interruption of the services provided. *Security* also has to be analyzed from a new perspective. Fog devices face threats that are not present in more controlled Cloud scenarios, regarding both *security* and *privacy* [32]. Additional challenges have been identified [33] and include dealing with a *larger scale* of devices and a highly *dynamic* environment.

These challenges call for the make-up and development of new orchestration mechanisms for the Fog. These mechanisms are to be executed under the control of a well-designed orchestrator that combines the advantages of centralized and distributed approaches, for which an architecture is provided in the following section.

IV. A hybrid approach for Service Orchestration in the Fog

To cope with the Fog challenges described in Section III while still guaranteeing an efficient resource management, a service orchestrator architecture for the Fog using a hybrid approach combining orchestration and choreography is proposed in this section.

In order to fully understand the architecture in charge of the orchestration and choreography of services in the Fog, a complete review of the general scenario is provided. The scenario, depicted in Fig. 1, shows three levels: 1) IoT Level, 2) Fog/Cloudlet Level, and 3) Cloud Level. The Virtual Clusters are at the IoT Level; these are composed by the grouping of terminal communication devices (e.g. smartphones, vehicles) that communicate among each other and also with the devices belonging to neighboring Virtual Clusters. This allows the movement of the IoT devices granting a higher level of freedom for the users. Communication among Virtual Clusters is achieved by the usage of choreography mechanisms that allow a quick reaction to possible changes in the topology (e.g. movement from one Virtual Cluster to another) as well as providing continuity to the services and thus higher resilience. Furthermore, this approach also benefits real-time applications by allowing a faster response among the devices (shorter paths), without the need of intervention of another device at a higher level of the infrastructure.

Edge Communication Links and Edge Gateways enable the communication of the IoT and Fog/Cloudlet Levels. The Fog/Cloudlet Level is sub-divided in the South-Bound Region and North-Bound Region. The South-Bound Region, closer to the IoT, is composed of Fog Instances, or Cloudlets. Each Fog Instance is a set of Fog Computing Devices, that allow different actions such as the migration of services for processing from the IoT devices (code offloading); Fog Communication Devices, that allow the connection between the various levels of the infrastructure, and also among the Fog Instances via Access Points and Base Stations; and Fog Storage Devices, that enable caching of content for nearby Fog and IoT users. This last part is also achieved by using choreography mechanisms.

For the resource management at the North-Bound Region of the Fog/Cloudlet Level and Cloud Level, an orchestration approach is used. The communication between the North-Bound Region of the Fog and the Cloud is done via the Fog-Cloud Gateways and Fog-Cloud Links. The Cloud allows a massive amount of resources for heavy computation and storage requirements, usually known as Cloud Analytics. By using an orchestration approach at these levels, it is possible to have a global view of the system.

With the adoption of a hybrid approach, different advantages are achieved. For instance, it allows independence of the lower levels for their decision-making processes, which enables a quicker reaction in case of failures or topology changes, and also lower response time for time-constrained applications. Furthermore, in the upper levels, having a global view of

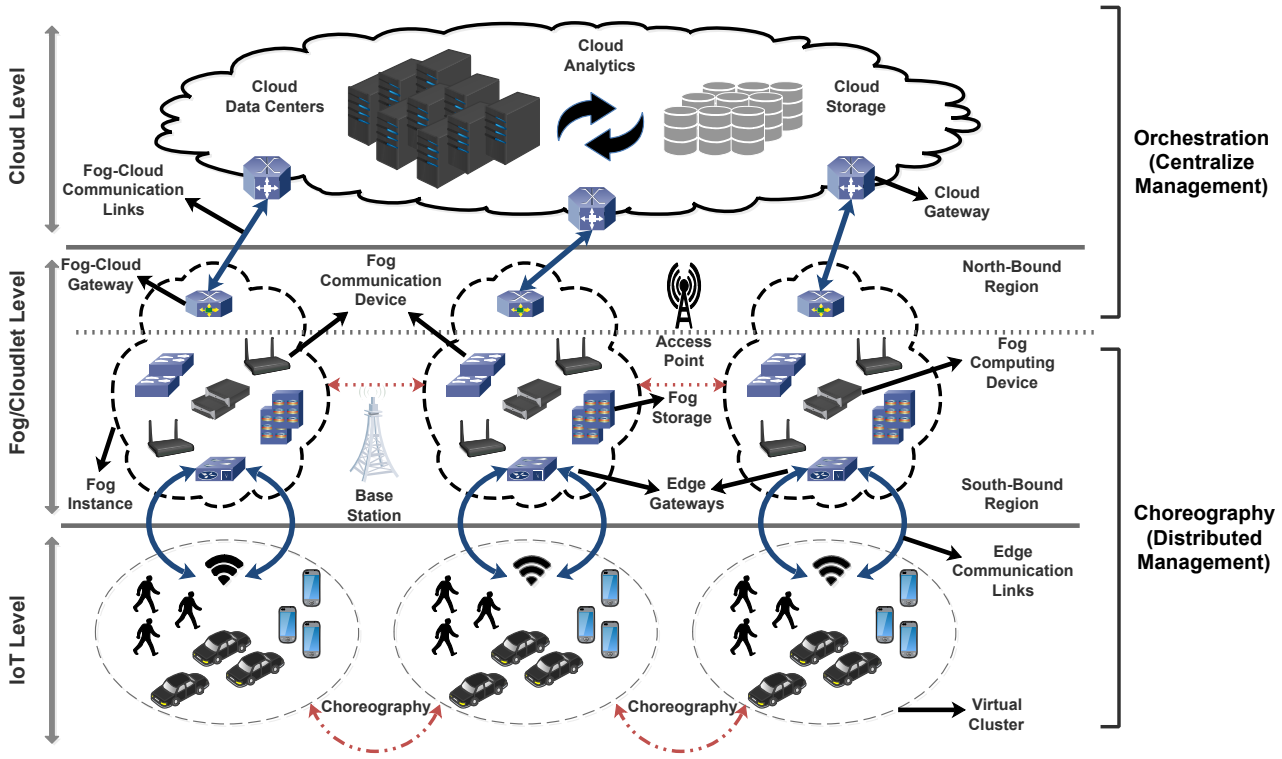

Fig. 1: Logical Network Infrastructure for Service Orchestration

the system allows applying long-term actions aiming at the optimization of the overall system.

To manage the resources and communication in the previously described scenario, an orchestrator architecture is proposed in Figure 2. Overlapped instances of this architecture ought to be replicated at different levels, namely at Fog Instances (Fog/Cloudlet Level) and Virtual Clusters (IoT Level) allowing the implementation of distributed choreography mechanisms; and at the Cloud Level, where a single logical instance is also deployed for global orchestration purposes.

The architecture is composed of different modules. The *Communication Manager* controls the communication among the different orchestrator instances, deployed in Virtual Clusters, Fog Instances, and the Cloud. The *Resource Manager* monitors the resource usage of the various devices, including which of these resources are being used (and by whom) and which are available or idle. The *Service Discovery* module allows lookups for services and applications that are available and where is the nearest instance being executed. It also allows the aggregation or removal of services at any time. The *Security Manager* provides different mechanisms for authentication and privacy, according to the applications and services requirements.

The *Status Monitor* supervises the different activities in the system. It takes the information from the Resource Manager

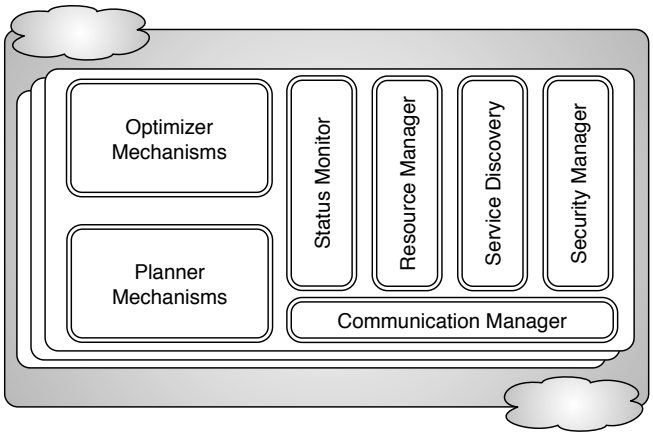

Fig. 2: Hybrid Orchestrator Architecture

to guarantee that the required QoS and QoE levels previously agreed by the users are being met. The *Planner Mechanisms* schedule the execution of processes throughout the system. They set up where and when each service and application is going to be executed.

Finally, the *Optimization Mechanisms*, only applied at the upper levels of the infrastructure where a global view is used, to improve the performance of the system and the QoS and QoE for users. Both the Planner Mechanisms and

Optimization Mechanisms can be instantiated according to the particular requirements of the applications and services running at the Fog Instance or Virtual Cluster. Thus, resilience-focused mechanisms can be preferred over real-time or security-focused mechanisms, according to different needs.

This architecture was designed in the frame of the SORTS project, which is described in the section below.

V. The SORTS Project

The *Supporting the Orchestration of Resilient and Trustworthy Fog Services* (SORTS) project, a cooperation between Portugal (University of Coimbra) and Brazil (University of Campinas), aims at the design, implementation, and evaluation of a service orchestrator for Fog environments, capable of guaranteeing resilience and trustworthiness for dynamic services in the Fog. The service orchestrator will have to take care of the composition of simple service elements (e.g. storage processing, sensing) into more complex services (e.g. trip planning) that can satisfy the user requirements in the Fog.

The orchestrator will also have to guarantee agreed levels of QoS even in the case where the different service elements are spread over the Fog, which can hamper several processes such as resource allocation. Also, mechanisms to ensure low-latency, resilience, and scalability have to be incorporated as functions of the designed service orchestrator.

Since the Fog can spread over a large geographic area, the service orchestrator will operate in a loosely coupled mode, with more real-time demanding functions controlled by the devices at a Virtual Cluster/IoT Level, following a choreographic approach, with a set of regions grouped in a Fog Instances/Cloudlets enabling autonomous operation of the regions using a choreography approach.

This way, the devices at the IoT and Fog/Cloudlet (South-Bound Region) Levels cooperate among each other freely to take low-level autonomic management decisions (micro-management); while at the Fog/Cloudlet (North-Bound Region) and Cloud Levels, a larger time scale management will take place (macro-management).

Under this approach, the IoT and Fog/Cloudlet (South-Bound Region) Levels will control events such as load variation, topology changes, delay variations, and real-time management; whereas the Fog/Cloudlet (North-Bound Region) and Cloud Levels will tackle the global optimal for the group of regions by keeping a global view.

The SORTS hybrid orchestration approach described thus far allows the execution of complex management actions required in Fog environments. To demonstrate its usage, an example based on Smart City mobility support is provided below.

A. An Architecture Instance for Mobility Support

The orchestrator presented in Section IV can be instantiated according to the requirements of the applications being executed in the different levels of the infrastructure. The orchestrators modules can be adapted to diverse needs to provide optimized solutions. This section describes the instantiation of the orchestrator to offer mobility support in a Smart City scenario.

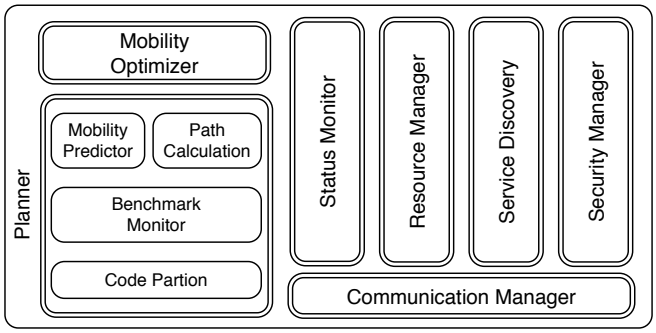

Fig. 3: Hybrid Orchestrator Architecture: An Instance for Mobility Support

Given the mobility of many devices in the IoT (e.g. connected vehicles, cyclists, and pedestrians), the Fog orchestrator must manage the Fog resources according to the demand required by such applications. Figure 3 shows the orchestrator instance that handles the mobility use-case. The Status Monitor, Resource Manager, Service Discovery, Security Manager, and Communication Manager modules remain the same, with the functionalities described in Section IV. The *Optimizer* is refined with mechanisms specifically focused on mobility as well as the *Planner*.

For this orchestrator instance, the Planner is composed of four specific mechanisms to support mobility. The *Mobility Predictor* deals with the precalculation of estimation regarding user mobility patterns. This estimation feeds the *Path Calculation* mechanisms that determine the best route between the mobile devices and Fog Instances/Cloudlets. The *Benchmark Monitor* works together with the Status Monitor to decide if the task offloading is actually needed, and if so, determines the optimal location for the services that belong to an application. To achieve this, the *Benchmark Monitor* evaluates the available resources both in the device and in the Fog Instances and also the expected QoS and QoE of applications and services. The *Code Partition* mechanisms determine which services are to be offloaded to the Fog Instance/Cloudlet.

B. Interaction between the Elements of the Architecture

To understand how these mechanisms work together, a sequence diagram is shown in Figure 4. Three actors are involved in this example: the Status Monitor, the Planner, and the VM/Container. The locally executed applications and services are monitored, and according to the resource demand can be migrated to a specific Fog Instance that guarantees the QoS and QoE requirements of applications and services. Once a request is received, the Planner must determine where the application (or parts of it) is going to be executed. The Status Monitor searches for information about the user device to determine if the application can be executed locally, or on

the other hand, if it must identify the nearest Fog Instance that represents the best option for the user. According to the resource demand from the applications and the resources available in the Fog Instance, the Planner should select which tasks will have to be migrated from the IoT device to the Fog.

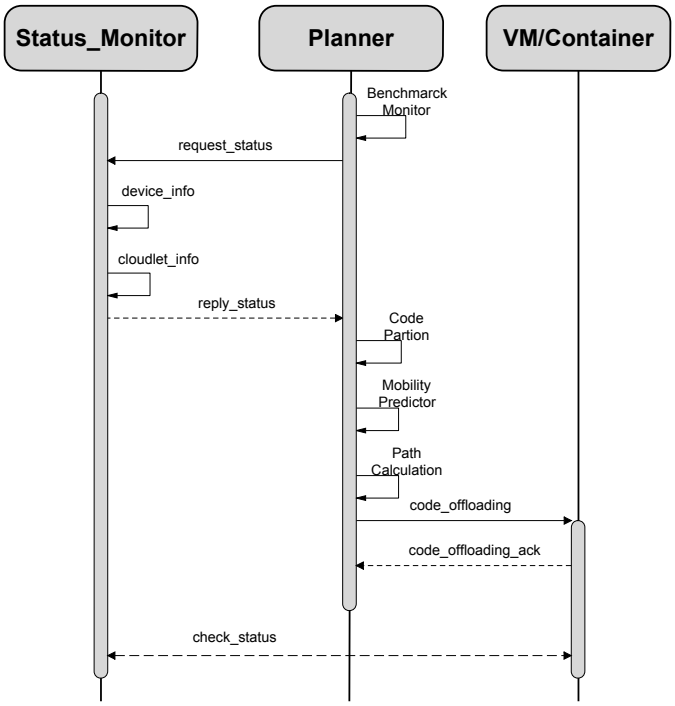

Fig. 4: Sequence Diagram: Code Offloading in a Mobility Scenario

Given the mobility of the devices, the application should preferably be executed in the Fog Instances that offer the best conditions to the user while he is moving. In this context, it can be required to migrate an application from one Fog Instance to another to offer the best execution conditions while the device is in movement. The planner can select the Fog instances to act in the path. In the selection process, the Planner uses the device mobility data (provided by the Status Monitor), trajectory predictions (from the Mobility Predictor mechanisms) and data obtained using choreography from other Fog Instances (e.g. location and available resources).

From a set of preselected Fog Instances, the Planner can define which Fog Instance best fits the user in a specific point of its future path (Path Calculation). Another functionality that can be inferred from such path is the possibility to minimize the amount of migration among the Fog Instances. The applications can be distributed between the Fog Instances that will be available for longer within the device's route, before needing a new migration. The next step is offloading the code to the VM located on the selected Fog Instance.

In a normal flow of execution, the Status Monitor will oversee the methods of applications running in a VM/Container. The Planner will keep in touch with the application in order to identify the methods that can be migrated to the Fog without compromising its execution. The Status Monitor will oversee, in real-time, the resource demand for each method selected by the Planner. The data about the application resource demand and the available resources, both locally and remotely, is evaluated by the Status Monitor module. If the resources available in the device satisfy the application demands, then it will be executed locally. On the other hand, if the application requires a vast amount of resources and the Fog Instance with which the device already established a connection offers enough resources, the Planner can conclude that offloading the application is the most beneficial procedure to the user.

After the process of offloading is completed, the application is executed in the Fog Instance inside of a VM/Container as long as this state is the most beneficial for the application. The Status Monitor registers data from the connection quality and idle resources in neighboring Fog Instances. This data helps the Planner to select the best location to place the application given the user movement in case of a needed migration.

This example is focused on the mobility support, as the other modules inside the orchestrator (e.g. Communication Manager and Resource Manager) will continue to carry out their basic tasks. Additionally, this example illustrates how easily the orchestrator can be adapted to handle specific use-cases inside a scenario as complex as a Smart City.

VI. Conclusion

The development and deployment of IoT devices enable the proliferation of new services and applications. Some of these applications require more resources (e.g. processing, storage) than what the IoT devices can provide. The Cloud paradigm offers a solution for some of these applications, but other applications and services are not particularly fit for the Cloud. The Fog extends the idea of the Cloud bringing resources to the edge of the network, closer to the final user, enabling lower latency levels, location awareness, and mobility support among other advantages.

The combination of IoT and Fog encompasses a highly complex scenario with a huge amount of different devices that must cooperate with each other. This requires effective orchestration mechanisms to guarantee the smooth performance of applications and services. However, mechanisms typically applied to the Cloud can not naturally be migrated to the Fog given its particular characteristics. This calls for the design and development of new orchestration mechanisms for the Fog.

In this paper, a hybrid approach to manage the Fog is proposed, using the combination of orchestration and choreography styles of managing for different regions of the infrastructure. The architecture outlined, framed in the SORTS project, enables a global view which allows general optimizations, and also automated dynamic reactions at the lower levels. Additionally, by using different instances of the

orchestrator, it is possible to apply different optimization goals according to the application's requirements.

As future works, optimization models to apply in the different modules of the orchestrator are going to be developed, such as service placement mechanisms aimed at low latency, and disjoint path computation-focused on high resilience. Furthermore, a real implementation of the orchestrator is going to be carried out and evaluated in a real scenario in the context of the SORTS project.

ACKNOWLEDGMENT

The work presented in this paper was partially carried out in the scope of the projects: ≪MobiWise: From mobile sensing to mobility advising≫ (P2020 SAICTPAC/0011/2015), co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI), European Unions ERDF (European Regional Development Fund); and SORTS, financed by the the CAPES - Coordenção de Aperfeiçoamento de Pessoal de Nível Superior ≪CAPES-FCT/8572/14-3≫ and by the FCT - Foundation for Science and Technology ≪FCT/13263/4/8/2015/S≫. This work was also co-financed by the DenseNet project ≪PTDC/EEISCR/6453/2014≫, which is a FCT/FEDER/COMPETE 2020 project.

NOTE

This is a preliminary version of the paper. The final publication is available at IEEE Xplore via http://ieeexplore.ieee.org/document/8114500

REFERENCES

[1] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[2] B. Jaeger, "Security orchestrator: Introducing a security orchestrator in the context of the etsi nfv reference architecture," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Aug 2015, pp. 1255–1260.

[3] A. Mayoral, R. Vilalta, R. Muoz, R. Casellas, and R. Martnez, "Experimental seamless virtual machine migration using an integrated sdn it and network orchestrator," in *2015 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2015, pp. 1–3.

[4] L. Velasco, A. Asensio, A. Castro, J. L. Berral, D. Carrera, V. Lpez, and J. P. Fernndez-Palacios, "Cross-stratum orchestration and flexgrid optical networks for data center federations," *IEEE Network*, vol. 27, no. 6, pp. 23–30, November 2013.

[5] R. Vilalta, A. Mayoral, R. Casellas, R. Martnez, and R. Muoz, "Experimental demonstration of distributed multi-tenant cloud/fog and heterogeneous sdn/nfv orchestration for 5g services," in *2016 European Conference on Networks and Communications (EuCNC)*, June 2016, pp. 52–56.

[6] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[7] B. Martini, D. Adami, A. Sgambelluri, M. Gharbaoui, L. Donatini, S. Giordano, and P. Castoldi, "An sdn orchestrator for resources chaining in cloud data centers," in *2014 European Conference on Networks and Communications (EuCNC)*, June 2014, pp. 1–5.

[8] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, Mar. 2017. [Online]. Available: https://doi.org/10.1109/MIC.2017.36

[9] S. Cherrier, Y. M. Ghamri-Doudane, S. Lohier, and G. Roussel, "Services collaboration in wireless sensor and actuator networks: Orchestration versus choreography," in *2012 IEEE Symposium on Computers and Communications (ISCC)*, July 2012, pp. 000 411–000 418.

[10] G. Pedraza and J. Estublier, "Distributed orchestration versus choreography: The focas approach," in *Trustworthy Software Development Processes: International Conference on Software Process, ICSP 2009*, Q. Wang, V. Garousi, R. Madachy, and D. Pfahl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 75–86. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01680-6_9

[11] S. Meng and F. Arbab, "Web services choreography and orchestration in reo and constraint automata," in *Proceedings of the 2007 ACM Symposium on Applied Computing*, ser. SAC '07. New York, NY, USA: ACM, 2007, pp. 346–353. [Online]. Available: http://doi.acm.org/10.1145/1244002.1244085

[12] T. Furtado, E. Francesquini, N. Lago, and F. Kon, "A middleware for reflective web service choreographies on the cloud," in *Proceedings of the 13th Workshop on Adaptive and Reflective Middleware*, ser. ARM '14. New York, NY, USA: ACM, 2014, pp. 9:1–9:6. [Online]. Available: http://doi.acm.org/10.1145/2677017.2677026

[13] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[14] 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*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[15] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Nov 2015, pp. 73–78.

[16] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. New York, NY, USA: ACM, 2015, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/2757384.2757397

[17] S.-C. Hung, H. Hsu, S.-Y. Lien, and K.-C. Chen, "Architecture harmonization between cloud radio access networks and fog networks," *IEEE Access - Special Section on Emerging Cloud-Based Wireless Communications and Networks*, vol. 3, pp. 3019–3034, Dec. 2015.

[18] P. Varshney and Y. Simmhan, "Demystifying fog computing: Characterizing architectures, applications and abstractions," in *1st IEEE International Conference on Fog and Edge Computing (ICFEC2017)*, May 2017.

[19] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[20] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, ser. MCS '12. New York, NY, USA: ACM, 2012, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/2307849.2307858

[21] Y. Jararweh, L. Tawalbeh, F. Ababneh, and F. Dosari, "Resource efficient mobile computing using cloudlet infrastructure," in *2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks*, Dec 2013, pp. 373–377.

[22] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: architecture and service modes," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 18–24, June 2015.

[23] Y. Gao, W. Hu, K. Ha, B. Amos, P. Pillai, and M. Satyanarayanan, "Are Cloudlets Necessary?" Carnegie Mellon University, School of Computer Science, Tech. Rep., Oct 2015.

[24] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*. Cham: Springer International Publishing, 2014, pp. 169–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-05029-4_7

[25] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and software architecture for fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 44–53, Mar 2017.

[26] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *2014 International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2014, pp. 464–470.

[27] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. C. M. Leung, "Lightweight management of resource-constrained sensor devices in internet of things," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 402–411, Oct 2015.

[28] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for iot," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, March 2015, pp. 687–694.

[29] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *2014 Australasian*

Telecommunication Networks and Applications Conference (ATNAC), Nov 2014, pp. 117–122.

[30] T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *CoRR*, vol. abs/1502.01815, 2015. [Online]. Available: http://arxiv.org/abs/1502.01815

[31] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, March 2015, pp. 105–110.

[32] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *2014 Federated Conference on Computer Science and Information Systems*, Sept 2014, pp. 1–8.

[33] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for iot services: Issues, challenges and directions," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, Mar 2017.