Orchestration and Monitoring in Fog Computing for Personal Edge Cloud Service Support

Florian Wamser[1], Chiara Lombardo[2], Constantinos Vassilakis[3], Lam Dinh-Xuan[1],
Paolo Lago[2], Roberto Bruschi[2], Phuoc Tran-Gia[1]
[1]*Institute of Computer Science, University of Würzburg, Würzburg, Germany*
[2]*CNIT - Genoa Research Unit, Genoa, Italy*
[3]*Ubitech Ltd, Research Department, Athens, Greece*
[1]florian.wamser|lam.dinh-xuan@informatik.uni-wuerzburg.de
[2]roberto.bruschi@cnit.it, chiara@tnt-lab.unige.it
[3]constantinosvassilakis@gmail.com

***Abstract*—To effectively support new cloud services and smart device accessibility, current Internet design must move away from its typical TCP/IP-based architecture and evolve towards new paradigms characterized by high degrees of freedom. In particular, Network Functions Virtualization and Software-Defined Networking have proven to be appropriate approaches to introducing edge-programmability. In this article we give an overview over the fog computing-architecture approach INPUT (In-Network Programmability for the next generation of Personal CloUd Service SupporT) with its implementation OpenVolcano, which uses the above approaches to promote future personal Internet cloud services at the network edge. Specifically, it enables computing and storage capabilities for edge network devices, provides a simple, private, programmable SDN-based Ethernet domain for end users and edge cloud services, IaaS and PaaS support with tight delay and performance constraints through templates. The focus of this work is on multi-service orchestration and monitoring, which is an essential core of any edge computing architecture.**

***Index Terms*—**

I. Introduction

With the development of new communication technologies and applications, along with a large number of versatile end-user devices, there is a steadily growing demand for even more advanced services and applications that offer high Quality of Experience (QoE) for the end user. As a result, stringent requirements are posed for latency, bandwidth, storage and computational power. As more and more services are entering the cloud and data centers, new network and IT architectures are required that are both energy efficient and cost effective, while at the same time taking into account the two most important trends that drive network evolution: Network Functions Virtualization (NFV) and Software Defined Networking (SDN).

The recently concluded H2020 Project INPUT [1] has brought forth an innovative framework, which exploits in-network programmability capabilities for off-loading, virtualization and monitoring. To the goal of introducing computing and storage capabilities to edge network devices to move such virtual images closer to end-users, INPUT has devoted a lot of effort in the design of a number of control plane mechanisms for (physical and virtual) resource management, in order to avoid pointless network infrastructure and datacenter overloading and to provide lower latency reactiveness to services.

INPUT aims to virtualize the functionalities, capabilities, and content of physical smart devices and provide them as cloud services closer to the end users, thus improving user experience. This goal is used, on the one hand, by utilizing NFV to control the access and edge network and, on the other hand, by implementing new cloud architecture solutions with the help of SDN. To this end, new cloud paradigms are designed and structured along with network capabilities to provide support for new personal cloud services including their communications and information exchange. Among the architectural and logical extensions, monitoring, orchestration and consolidation in the cloud area play a central role.

This paper deals with the definition of the workflow for the orchestration/monitoring of the network services, with respect to user and energy requirements, and the description of the first solutions for the consolidation and optimization of networks in the cases of single and multi-service scenarios. We define metrics and algorithms based on different methods and ideas. Specifically, this work deals with the following two points:

- Definition of monitoring specifically designed with respect to the requirements of QoE, resource utilization and energy consumption.
- Orchestration approaches are presented in two directions, focusing on placing Service Apps or Network Functions close to the user, while ensuring QoE and low energy consumption.

The rest of the paper is structured as follows. Section II describes the application scenario that is addressed with the INPUT architecture. Subsequently, the INPUT architecture and, specifically, its control plane are described in Section III. In Section IV, the terms monitoring, orchestration and consolidation are defined in the context of the INPUT fog architecture. The results and performance measurements are given in Section V while Section VI concludes the paper with a summary of the most important findings.

II. Application Scenario

The INPUT Project was conceived to design a novel infrastructure for supporting Future Internet personal cloud services. The main goal was to move beyond classical service models (i.e., IaaS, PaaS, and SaaS) by providing users with the "virtual images" of their physical smart devices, which are deployed in a cloud/fog/edge environment with the advantages of additional scalability, sustainability and innovative added-value capabilities.

The INPUT platform is envisaged for deployment in the edge facilities of telecom operators, composed of computing and storage appliances interconnected by physical/virtual OpenFlow switches. The INPUT architecture embraces the edge computing paradigm, with full and state-of-the-art technological convergence of mobile and wireline access with the softwarization of network services, based on the NFV [3] and SDN [4] technologies and integrated with 4/5G mobile networks. To this goal, two VNFs have been developed to extract the control information needed for MEC attach points management and then inject and retrieve packets between user equipment and MEC data-plane. In this way, access and/or core network segments can be directly exposed to vertical applications in 4G legacy environment just line in 5G.

The personal services designed in the INPUT use cases (of which an example can be found in [5]) are designed in the form of a service chain that is stored in a service template managed by the INPUT platform [6] and instantiated only upon user subscription. The service chain is composed of individual service applications running in an execution container, which is typically a Virtual Machine (VM). Communication and information exchanged among applications of the same service chain, and between user and service instance, are handled through SDN overlays.

Moreover, service chains are characterized by proximity classes, which represents the maximum allowed distance between the user and his/her instances that guarantees the fulfillment of the Service Level Agreement (SLA). The appropriate placement and dynamic (re)allocation of the service chain with respect to the user position or is constrained by the proximity class as will be described in Section IV.

III. The INPUT Control Plane Architecture

The central element in the design of the INPUT platform is represented by the OpenVolcano open-source project [7], a software platform that exploits in-network programmability capabilities for off-loading, virtualization and monitoring to provide scalable and virtualized networking technologies able to natively integrate Edge Computing functionalities.

In the design of the INPUT platform, building blocks from OpenVolcano, along with the external support of the Ericsson Network Manager (ENM), have provided the control and the management processes of the infrastructure: the Network and Service Service Operating System (NS-OS) and the Network and Service Management (NS-MAN). Their main roles and interactions can be seen in Figure 1.

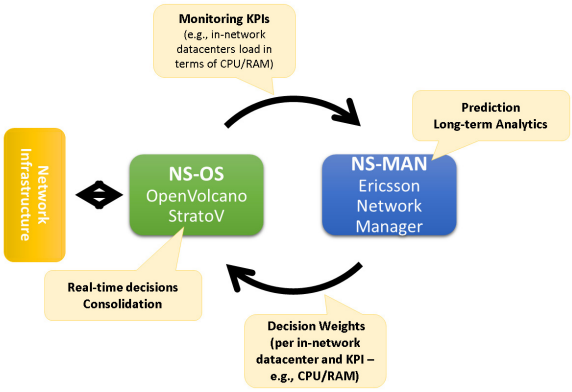

Fig. 1. Logical view of the INPUT control plane.

The NS-OS drives the real-time configuration of the programmable resources and the dynamic instantiation and migration of Virtual Network Functions (VNFs) and VMs composing the service chains according to users locations. In more detail, the NS-OS performs the following three main tasks: Consolidation, Orchestration and Monitoring.

The Consolidation task is in charge of calculating the optimal re-configuration of the infrastructure (e.g., the topology of the Personal Networks and the matching and action rules of the SDN switches) in terms of network paths/overlays and of the service chain instances locations, with the objective to match the required QoE/QoS and the estimated workload/traffic volumes with the minimum possible level of energy consumption. In particular, consolidation criteria will be designed to obtain the re-configuration matrix of the network infrastructure and of services.

The Orchestration mechanism takes the re-configuration matrix coming from the consolidation process as an input and instantiates/migrates service instances to the identified subset of devices/hardware resources, by changing the network configuration accordingly, without causing any service interruption or performance degradation.

In order to assure that the consolidated configuration works as expected and to support the fine tuning of future re-configurations, the monitoring sub-system will collect performance measures and alerts, which include network-, app-, and power-aware performance indexes.

The NS-MAN is responsible for the long-term configuration of the network, the administrative configuration of the infrastructure, the overlaying cloud services and personal networks. In addition, it is in charge of monitoring faults, intrusions and attacks in the system and uses trend analysis to predict errors and guarantee that deployed services are always available.

To do this, the NS-MAN stores historical data received from the NS-OS and the network and computing elements (e.g., network/servers usage, users mobility, service chains computational resources, etc.), for performing data analytics and providing trend estimates. The goal of these correlations is to get insights on the overall operating behaviour of network devices and computing facilities to predict service demand,

plan the resource provisioning, prevent congestion, possible failures, and maximize energy saving.

IV. Monitoring, Orchestration and Consolidation

We go into detail in this section about the relationships of monitoring, consolidation, and orchestration, which is important for the further understanding of this paper. The necessary mechanisms for the dynamic provision of resources and energy management are listed and justified within the objectives[1]:

- A Monitoring Sub-System to collect performance measures and alerts, which include network-, app-, and power-aware performance indexes.
- Orchestration Mechanisms to dynamically migrate in-network Apps without causing any service interruption or performance decay.
- Consolidation for the re-configuration of the smart infrastructure to meet the estimated workload and user/service requirements with the minimum possible level of energy consumption.

1) Monitoring: Monitoring refers to statistical information gathering from objects concerning, but not limited to, the performance of applications, communicating between them, physical utilization rates and power consumption.

The Apps-aware monitoring function provides system administrators the information of connected applications regarding their overall performance/state, quality and current resource utilization. For instance, a video quality monitoring function carries out the information about current video buffer, video resolution at the client and, possibly, the probability of stalling occurrence based on downstream bandwidth, which can be used to estimate the QoE level perceived by the users. In the INPUT architecture, DC_Apps and Service_Apps are monitored continuously at the server side, the User_Apps are monitored by dedicated Net_Functions that are deployed at the client side or in the INPUT network.

The Network-aware monitoring function, on the other hand, assesses the communication between applications, as well as various network components. This function is in charge of detecting compromised network components or potentially harmful events, such as an overload of traffic to servers or virtual objects, or failures in SDN controllers or OpenFlow switches. A detection of low bandwidth in the users access network can activate the migration process of applications/services to the edge.

The power-aware monitoring function is the enabler of interfaces to provide the current energy state of servers/network devices. Power consumption measurements can be called with a certain time periodicity. In the INPUT architecture, the Geyser module based on the ACPI standard monitors continuously the power consumption of an OpenVolcano server.

2) Orchestration: Since the INPUT architecture extends SDN and NFV technologies to support advanced network functionalities, virtual resources allocated to Service Apps or Network Functions must be automatically managed and orchestrated. To this end, an orchestration algorithm is designed and implemented in two steps: first, by an initial placement, and afterwards by a dynamic reallocation of resources. During the initial placement, the amount of resources required for a created task are estimated and provisioned. There, the goal is to satisfy the predefined Service-Level-Agreements (SLA), especially the QoS and QoE of all customers with the available resources at minimal costs. The biggest challenge is to allocate the available resources fairly and to make sure no customer is under-provisioned. Additionally, unutilized resources are shut down to save energy. The second phase in the orchestration process is the reallocation. There, a major challenge is to dynamically react on changing resource demands without causing any service interruptions or performance delays. The resource utilization can either be checked periodically or in any other timing pattern by the orchestration algorithm or upon the notification of the under-provisioned VMs. Afterwards, the reallocation of resources is started. The first option in adapting to changing requirements is adding resources to a running task, for example, adding RAM to a running virtual machine. The benefit of this method is that no task migration is necessary. The alternative method is the migration of a task to another location with higher available resources. Next to an iterative improvement of the allocation, where resources are allocated only to the under-provisioned task, a global optimization is possible. There, resource utilization of the whole infrastructure is monitored and unused resources are reallocated to other processes or shut down to safe energy.

3) Consolidation: Consolidation in cloud computing refers to the process of distributing the workload to a smaller number of servers. This is achieved by redirection of service requests to other service instances or the (live) migration of virtual machines (VM). Thereby, the surplus servers can be suspended or terminated, and the remaining servers can be better utilized. The ultimate goal is to save energy costs, thereby not violating service level agreements (SLAs), but still achieving a high QoE for end users. The consolidation problem is typically modeled as a vector bin packing problem, which is NP-hard [8]. Thus, heuristics are used to obtain solutions, such as first fit decreasing (FFD) and best fit decreasing (BFD) [9], [10]. However, a nave consolidation, which just puts the workload to the smallest number of servers and reaches close to 100% utilization can even increase the energy consumption. Thus, consolidation has to consider the aggregated resource consumptions of collocated VMs as well. Typically, resource consumptions are not additive, and a thorough understanding of the implications of collocated VMs is required to decide which VMs to consolidate on which server [5]. Additionally, the migration of running VMs between servers also has to be considered when consolidating VMs. It can cause downtime or reduced performance of the service, which negatively influences the QoE of end users. Thereby, the transmission of the VMs can even affect other services, which also share the same network links. A further challenge of consolidation is the prediction of changing workloads. If very elastic VMs

[1] http://www.input-project.eu/index.php/about/objectives

(in terms of resource requirements) are consolidated on a highly utilized server, there might be no possibility to scale up the resource allocations if the workloads increase. This results in reduced performance until new VM instances can be started or the VM is migrated to a server, which can fulfill the increased requirements. Note that this might necessitate to power on further servers. In the context of the INPUT project, consolidation refers to operating all requested services and their corresponding Service_Apps with a minimum number of INPUT nodes. The consolidation process can be triggered by the NS-MAN either periodically (e.g., at regular time intervals), or it can be event-based (e.g., when the number of services in the INPUT system changes), or threshold-based (e.g., if the resource utilization of an INPUT node drops below a threshold). Based on the requirements of the currently running services and the resource utilization of the INPUT nodes, which are obtained from the monitoring, consolidation actions are decided and implemented. The consolidation actions include the migration of Service_Apps, as well as the redirection of requests to newly instantiated Service_Apps. Subsequently, all emptied INPUT nodes will be suspended or terminated.

V. Evaluation

Cloud infrastructure optimization requires efficient cloud orchestration algorithms and procedures and is tightly related to workloads consolidation within the cloud infrastructure. The ultimate goals are serving the highest possible QoE for the user and saving energy on the infrastructure side.

In the INPUT cases, a significant factor for high QoE is the appropriate placement and dynamic (re)allocation of Service_Apps of the service chains (as close to the user or to the previous Service_App as possible) and their termination when necessary. In this context, especially considering Edge Computing capabilities and Service Chains, initial placement of Services_Apps shall make use of specific algorithms that take into account service apps specific requirements such as Service Chain sequence, location/proximity to end user location, performance of Service_apps, availability of cloud infrastructure nodes/hosts, etc. Although this Service Chain Placement problem presents similarities to Virtual Network Embedding (VNE) and Virtual Data Center Embedding (VDCE) problems, a new problem formulation is required to include the full extent of parameters that reflect the aforementioned requirements.

A. The Service Chain Placement Problem

A service chain consists of several distinct interconnected software components. The deployment of a service chain over an infrastructure requires assigning an execution environment to each software component, as well as an overlay link routed over one or more physical links for each communication channel between two interconnected software components. The infrastructure consists of several physical machines able to host several execution environments (i.e. hypervisors running on physical machines hosting several virtual machines) and several physical links able to route over them several overlay channels. Hosts and links offer resources of certain capacity (capacitated resources). Indicative resources for hosts are CPU, memory and storage I/O rate, while links most commonly offer capacitated bandwidth resource. An execution environment for an application software component requires a certain amount of all or a subset of resources offered by a host. Meanwhile a channel between software components requires all or a subset of resources offered by a link assigned for routing. Assignments may require not only a certain amount of a capacitated resource but as well a certain monitored metric to be within a certain range, e.g. link delay. These requirements, along with others, specify what is considered a feasible assignment and form the constraints of the problem of deploying a service chain over the infrastructure.

B. Formulation of the Service Chain Placement Problem

In the following, an approach to formulating the optimization problem is given. In this approach, (1) host assignment and overlay path building are considered at a single phase providing for better solutions to the problem; (2) the cost initially considered is the distance measured in physical link hops of the overlay links that is created after the assignment of the communication links (channels) between application components; (3) capacitated resources and metrics constraints are considered; (4) placement is considered for the overall service chain considering that the user application and shared service apps are statically placed at the time of the placement problem solving; thus, service application components placement is decided for all except those already placed; (5) DC applications may be considered with this general formulation as statically placed components, as well; (6) the output after solving the problem is the matrix indicating where each component is placed and the matrix indicating which links are used to build overlay paths between components; (7) for collocated components, a path is not decided, since the connection is represented internally within the host.

In the formulation of the optimization problem, the following notations are used:

TABLE I
Summary of Parameters Used in the ILP Model

Parameters	Description
T	Set of application components
C	Set of channels between application components, $C \subseteq T \times T$
H	Set of hosts
L	Set of links between hosts, $L \subseteq H \times H$
S	Set of user application components statically allocated at hosts, $S \subset T$
H^S	Set of hosts where static user application components are placed, $H^S \subset H$, $f : S \to H^S \mid \forall h' \in H^S, \exists t' \in S : h' = f(t')$ (f is subjective)
R	Set of unique resources offered by hosts
R'	Set of unique resources offered by links
M	Set of monitored metrics at hosts
M'	Set of monitored metrics at links
a_t^r	Amount of resource r demand by application component t
i_h^r	Capacity of resource r at host h
β_h^r	Amount of resource r available at host h
m_h^k	Measured value of metric k at host h
c_{sd}^r	Amount of resource r demand required by channel (s, d)
b_{uv}^r	Amount of resource r available at link (u, v)
μ_{uv}^k	Measured value of metric k at link (u, v)

Based on the notations and the considerations mentioned before, the optimization problem is formulated as follows.

Given:

$$R = \{CPU, Memory\}$$
$$R' = \{Bandwidth\}$$
$$M = \emptyset$$

Minimize:

- ***Objective 1:*** Minimize total delay of service chain

$$Objective_1 = \sum_{(s,d)\in C} \pi_{uv,sd}\mu_{uv}^{Delay}, \quad \forall (u,v) \in L. \quad (1)$$

- ***Objective 2:*** Minimize resource utilization

$$Objective_2 = \sum_{h\in H} (min\{\sum_{t\in T} \sigma_{ht}, 1\} \frac{100\beta_h^{CPU}}{i_h^{CPU}}). \quad (2)$$

Objective 2 aims to minimize the product of the number of servers used in a placement and the percentage of available CPU. This optimization procedure will attempt to collocate VMs in a server but will have preference to an already utilized server. Thus, it will try to leave in a placement unused those servers that have zero utilization. By doing this, they may be put in idle state to save energy. However, at the initial placement all servers will have the same probability to be selected. The objective function does not favor any of them (e.g., a server with high capacity).

Subject to:

$$\sigma_{ht} \in \{0,1\}, \quad h \in H, t \in T. \quad (3)$$

In constraint (3), σ_{ht} is a decision variable and equals to 1 if task t is assigned to host h, 0 otherwise.

$$\sum_{h\in H} \sigma_{ht} = 1, \quad \forall t \in T, \quad (4)$$

$$\sigma_{ht} = 1, \quad h \in H^S, t \in S, \quad (5)$$

$$\sum_{t\in T\setminus S} \sigma_{ht} = 0, \quad \forall h \in H^S. \quad (6)$$

Constraint (4) ensures that a task (or application component) is assigned only to one host. The static placement of the user task is defined in Eq. (5), it is given as an input to the problem and not decided. Whereas, constraint (6) specifies that user applications are only placed in hosts assigned for them.

$$\sum_{t\in T} \sigma_{ht}\alpha_t^r \leq \beta_h^r, \quad \forall r \in R, \forall h \in H. \quad (7)$$

Equation (7) stipulates that the considered host h must have enough resources to allocate the application component t.

$$\pi_{uv,sd} \in \{0,1\}, (s,d) \in C, (u,v) \in L. \quad (8)$$

In Eq. (8), $\pi_{uv,sd}$ is a decision variable and equals to 1 if task channel (s,d) is routed from link (u,v), 0 otherwise.

$$\sum_{(u,h)\in L} \pi_{uh,sd} + \sigma_{hs} = \sum_{(h,v)\in L} \pi_{hv,sd} + \sigma_{hd}. \quad (9)$$

Constraint (9) captures and expresses in one equation,

- the unsplittable flow constraint: A channel uses a single outgoing link from source and a single incoming link at destination and does not split,

$$\sum_{(u,h)\in L} \pi_{uh,sd} = 1 \quad if \quad \sigma_{us} = 1,$$
$$\sum_{(h,v)\in L} \pi_{hv,sd} = 1 \quad if \quad \sigma_{vd} = 1,$$

- the collocation of tasks: A communication path is not required in the case that both s and d are assigned to the same host (and no capacity checking),

$$\sigma_{hs} = \sigma_{hd},$$
$$\pi_{uu,sd} = 0,$$

- the flow conservation constraint: No traffic is stored in a node unless this node is the source or the destination or collocated source and destination,

$$\sum_{(u,h)\in L} \pi_{uh,sd} = \sum_{(h,v)\in L} \pi_{hv,sd},$$
$$\forall h \in H : \sigma_{hs} = 0, \sigma_{hd} = 0.$$

$$\sum_{(u,h)\in L} \sigma_{hs}\pi_{uh,sd} = 0. \quad (10)$$

Constraint (10) makes sure that there is no loop in the path before reaching destination.

$$\pi_{uv,sd} = \pi_{vu,ds}, \quad (s,d),(d,s) \in C, (u,v),(v,u) \in L. \quad (11)$$

As determined in Eq. (11), a bidirectional communication between two tasks is routed through the same bidirectional overlay path. In addition to this, upstream and downstream of a flow is not routed separately.

$$\sum_{(s,d)\in C} \pi_{uv,sd}c_{sd} \leq b_{uv}, \quad \forall (u,v) \in L. \quad (12)$$

Constraint (12) guarantees that the link (u,v) must have enough resource required by channel (s,d).

C. Initial Performance Evaluation

In this Section, we present an initial performance evaluation of the Vent and Crater modules of OpenVolcano, which implement the service placement and the MCO procedures, respectively. In particular, we stress the orchestration mechanism to evaluate the time required to calculate the service positions and the OpenFlow rules to create the overlay networks.

Considering the complex operations performed by the orchestration algorithms, our objective is to assure that the proposed procedures are able to orchestrate virtualized services without compromising the KPIs defined in [10] related to the time to deploy new services and the migration times when the user moves.

We run the OpenVolcano orchestrator on a Linux server equipped with a hyper-threading- enabled Intel R Xeon R E5-2620 v4 2.10GHz processor [42], and we emulated a real wide-area topology obtained from the datasets available in [41]. In more detail, we adapt the Interroute topology to consist of 20 randomly selected datacenter nodes and 90 transit/access nodes, interconnected by 148 edges.

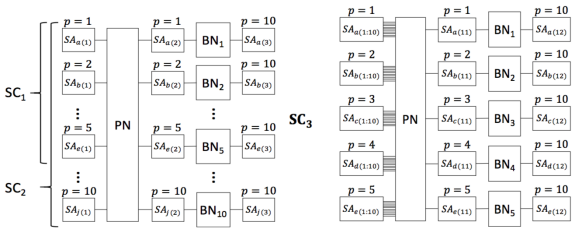

Fig. 2. Template of the testing service chains.

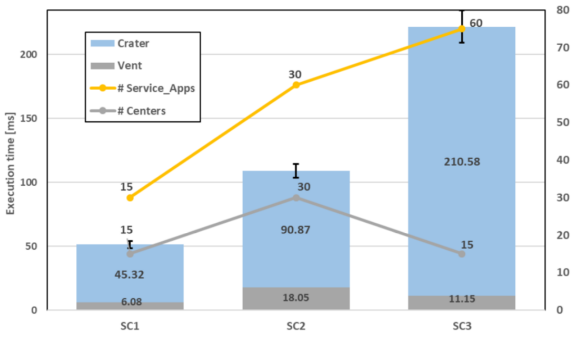

Fig. 3. Service placement execution time for the Crater and Vent modules according to the subscribed service chain.

We considered three service chaining scenarios SC1, SC2 and SC3, as illustrated in Figure 2. These SCs are designed to cover variations in the number of centers and proximity levels (p) and Service_Apps involved. Moreover, it can be observed that they also take into account possible interconnections between Service_Apps and Back-end Networks.

To measure the time required for calculation of the Service_Apps/Centers positioning and generation of the OpenFlow rules when the user subscribes to a new service, we emulated 100 service subscriptions for each of the service chains, with random CPU and RAM requirements.

Figure 3 shows the obtained calculation times of the Crater and Vent modules according to the different number of Service_Apps and centers of the three service chains in Figure 2. As we can observe, the most time spending task is the creation of the MCO network, which reaches 210 ms for the most complex service chain (SC3) composed of 60 Service_Apps. Differently, the time required for running the SSPP in Vent is almost negligible and it mainly depends on the number of centers to be placed. Indeed, as previously anticipated, the MCO mechanism allows reducing the complexity of the placement policy, which reaches a maximum calculation time of 18 ms.

These results demonstrate that the orchestration policies implemented introduce negligible delays in the time required to deploy and migrate services. However, in real scenarios further delays can be introduced by the time required by services to be ready to be used, which mainly depends on the boot time of the virtual machines and their configurations.

VI. Conclusions

In this work we describe the INPUT fog computing architecture. To efficiently and economically support new cloud services and edge devices, the Internet emerges into an architecture characterized by high degrees of freedom. The INPUT architecture is a consistent fog computing implementation, which exploits in-network programmability capabilities for off-loading, virtualization, and monitoring. Physical and virtual resource management control plane mechanisms provide the ability to move computing and storage capabilities to the end user to enable short service response times. Physical smart devices are integrated into the architecture as virtualized images.

The paper describes the orchestration, monitoring, and consolidation workflow for services, taking into account user and energy considerations. For this, service chains for services are defined and the service chain problem is specified and formulated. The paper concludes with a performance analysis of three different service chain scenarios. The implemented policies result in negligible delay when the service is instantiated or migrated. Future work includes the quantification of the delay for arbitrary services with different boot times and configurations within the INPUT and other fog architectures.

References

[1] The In-Network Programmability for next-generation personal cloUd service supporT (INPUT) Project, URL: $http://www.input-project.eu/$.

[2] R. Bruschi, G. Burgarella, P. Lago, "A Lightweight Prediction Method for Scalable Analytics of Multi-Seasonal KPIs," Communications in Computer and Information Science, 766: 61-70, 2017. DOI: $10.1007/978-3-319-67639-5\_6$.

[3] M. Chiosi et al., "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges and Call For Action," In Proceedings of the SDN and OpenFlow World Congress, Darmstadt, Germany. ETSI White Paper. URL: $https://portal.etsi.org/nfv/nfv\_white\_paper.pdf$

[4] Open Network Foundation (ONF), "OpenFlow Switch Specification," Version 1.4.0, URL: $https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf$.

[5] R. Bruschi, F. Davoli, P. Lago, A. Lombardo, C. Lombardo, C. Rametta, and G. Schembra, "An SDN/NFV Platform for Personal Cloud Services," IEEE Transaction on Network and Service Management (TNSM) vol. 14, no. 4, Dec. 2017, pp. 1143 - 1156.

[6] R. Bruschi, G. Genovese, A. Iera, P. Lago, G. Lamanna, C. Lombardo, and S. Mangialardi, "OpenStack Extension for Fog-Powered Personal Services Deployment," Proc. of the First International Workshop on Softwarized Infrastructures for 5G and Fog Computing (Soft5 2017), Genoa, Italy

[7] R. Bruschi, P. Lago, G. Lamanna, C. Lombardo, S. Mangialardi, "OpenVolcano: An Open-Source Software Platform for Fog Computing," Proc. of the 1st ITC Workshop on Programmability for Cloud Networks and Applications (ITC PROCON 2016), Wrzburg, Germany, Sept. 2016.

[8] Y. Dong, L. Zhou, J. Chen, B. Zheng, and J. Cui, Energy Efficient Virtual Machine Consolidation in Mobile Media Cloud. In: Picture Coding Symposium (PCS), pp. 248-252, IEEE, 2015.

[9] A. Beloglazov, J. Abawajy, and R. Buyya, Energy-Aware Resource Allocation Heuristics For Efficient Management Of Data Centers For Cloud Computing. In: Future Generation Computer Systems, vol. 28, no. 5, pp. 755768, 2012.

[10] A. Beloglazov and R. Buyya, Energy Efficient Resource Management In Virtualized Cloud Data Centers. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 826831, IEEE Computer Society, 2010.