Experimenting SDN and Cloud Orchestration in Virtualized Testing Facilities: Performance Results and Comparison

B. Martini, M. Gharbaoui, D. Adami, P. Castoldi, S. Giordano

***Abstract*—Due to the impressive demand for communication-intensive applications deployed in the cloud, cross-layer orchestration solutions for Cloud data centers (DCs) are essential for the effective usage of both network and cloud resources while addressing service requirements and user expectations. In this regard, the Software-Defined Networking (SDN) can play a key role thanks to programmable network control operations and to a more effective inter-working between network controllers and cloud management platforms. On the other hand, due to the relevant size of Cloud DCs, a significant evaluation of SDN-based orchestration solutions requires testing environments with proper scales and significant level of fidelity.**

This paper discusses the opportunities of using virtual testbeds to carry out Cloud-related experimentations while addressing with significant scale requirements of DC infrastructures. Firstly, we present a set of results of a test campaign we carried out using the Fed4FIRE experimental facility aiming at a comprehensive performance evaluation of an SDN-based orchestration solution for Cloud DCs. Secondly, we provide a comparison between virtual testbeds and other testing environments, i.e., laboratory testbeds, simulators and emulators, in terms of offered advantages and effectiveness in experimentations. Moreover, we discuss the appropriateness and advisability of using virtualized testbeds for research on novel SDN and Cloud orchestration solutions at scale, by highlighting advantages of virtual testbeds in comparison with laboratory, simulation and emulation testing facilities in terms of effectiveness of experimentations and accuracy of results.

***Index Terms*—SDN, Cloud Computing, Data Center, Virtual Machine, Orchestration, Virtual Testbed**

I. Introduction

THE increasing popularity of Cloud Computing applications and the prominent advances in virtualization software technologies (e.g., Hypervisor and Virtual Machines (VM)) have driven Data Center (DC) infrastructures toward greater complexity and workload dynamicity. The large amount and the high volatility of VM deployments give rise to unpredictable traffic patterns that might have an impact on the proper operation of the DC network due to the over-subscription of switches and possible bandwidth contentions. To meet this challenge, Cloud DCs require more advanced network control capabilities toward the automation of service deployments and a tighter coordination (i.e., orchestration) with cloud resource management operations [1] [2].

B. Martini, M. Gharbaoui, D. Adami are with CNIT, Pisa, Italy - {barbara.martini, molka.gharbaoui, davide.adami}@cnit.it; P. Castoldi is with Scuola Superiore Sant'Anna, Pisa, Italy - castoldi@santannapisa.it; S. Giordano is with University of Pisa, Pisa, Italy - stefano.giordano@iet.unipi.it;

The Software Defined Networking (SDN) paradigm promises to foster innovation in DC networks especially addressing the above requirements [3]. Indeed, SDN can effectively provide programming abstractions that can be exploited via software network controllers to dynamically enforce traffic steering rules into switches and to set-up data delivery paths while addressing specified requirements (e.g., bandwidth demand). In addition, software network controllers may offer northbound interfaces that can be exploited to effectively put into operation orchestration tasks to assure both traffic and computing demands of cloud applications as result of a combined control of both DC network and cloud resources [4] [5].

As an approach to address these challenges, we designed and developed an SDN-based orchestration system for Cloud DCs, i.e., SDN-DC orchestrator, able to interwork with both an SDN controller and a cloud management platform (i.e., VM Manager/Hypervisor) to adaptively coordinate and automatically provide both network and computing services while satisfying specified service requirements (i.e., CPU core and bandwidth demands). The resources underpinning the provisioned services are selected based on their availability (i.e., current load) evaluated from data status collected by network devices. We deployed and run the SDN-DC orchestrator as part of a EU project promoting the set-up of SDN-based European experimental facility [6]. Within this project, we focused on (i) the specification of orchestration algorithms for the coordinated and adaptive selection of computational and network resources [7], (ii) the design, development and experimental validation of an SDN-DC orchestrator prototype [8]. To validate the orchestration approach, we firstly relied on a simulator to assess the devised algorithms using a significant number of network and compute nodes to assess performance indicators, such as blocking probability of service requests and the rate of resource utilization. Secondly, we set-up a laboratory testbed reproducing a Cloud DC using virtualized servers and we carried out validation tests to verify the orchestrator software program against system requirements and to assess additional performance indicators (e.g., service set-up time).

However, based on these testing activities, we realized that a significant evaluation of SDN-based orchestration system for Cloud DCs requires experimental environments with a larger scale and with a higher level of fidelity to reproduce a testing environment as close as possible to production DC environments where the SDN-DC orchestrator would be even-

tually put into operation. Indeed, although relying on software virtualization (i.e., hypervirsor), by using a laboratory testbed we could not deploy a testing environment with an adequate size to achieve accurate performance results. On the other hand, by simulations we could not assess a comprehensive set of orchestrator performance indicators (e.g., the actual server and link utilization, rate of data throughput degradations) since the system operation (e.g., network data plane) is just synthetically reproduced. Hence, another challenge to foster innovation in Cloud DCs is to promote an adequate quality of experimentation using testing facilities with proper scales and significant level of fidelity. These features are especially relevant for cloud and DC experimentations where the number of involved systems (i.e., servers, switches), processes (e.g., cabling), and service dynamics are intrinsically high.

To address the aforementioned challenges, in this paper we present and discuss experimental results collected during a long-running test campaign carried out in a virtualized testbed facility to assess a resource orchestration solution for Cloud DCs. Virtualized testbed facilities are virtual laboratories built on top of a coordinated federation of testbeds according to the experimentation frameworks (e.g., FIRE [9]) to offer large-scale and wide range of technologies for researchers to remotely perform experiments in a variety of application fields (e.g., e-health). In the following, the terms virtualized testbed facility and virtual testbed are used interchangeably.

More specifically, we provide the following twofold contribution:

1) performance evaluation of the SDN-DC orchestrator based on experimental results we collect using a virtual testbed reproducing a DC facility on top of the Fed4FIRE experimentation infrastructure [10] assuring both a significant level of fidelity and a proper scale of experimentation so as to obtain meaningful performance data and complementary set of performance indicators. Indeed, Fed4FIRE platform is able to offer a large federation of testbed facilities featured by a wide range of Internet technologies (i.e., wired/wireless networks, Internet of Things, SDN, cloud computing), large system capacity (e.g., large number of computing nodes) and a rich set of experimentation services (e.g., bare metal services offering remote control of physical machines to experimenters).
2) discussion on the appropriateness and advisability of using virtual testbeds for experimentations on cloud and DC systems and a comparison of virtual testbeds with laboratory, simulation and emulation testing facilities in terms of effectiveness of experimentations and accuracy of results. This contribution is the result of a wide hand-on experience in testing the SDN-DC orchestrator in a variety of testing facilities. Indeed, we also provide a discussion of the obtained experimental results also in terms of soundness and consistency with performance results we previously obtained from laboratory experiments and simulations [7] [8] [11].

The paper is organized as follows. Section. II gives an overview of related works in the literature. Section. III describes the virtual testbed we set-up to carry out experiments on SDN-based orchestration in Cloud DCs and the features of the SDN-DC orchestrator prototype. It then reports the experimental results we collected using the virtual testbed. Section. IV presents the main advantages in using virtual testbeds along with a comparison among different testing environments in terms of offered capabilities and scales. It then discusses the experimental results against the ones we obtained while testing SDN-DC orchestrator in laboratory and simulation testing environments. Finally, Section. V provides the concluding remarks for this paper.

II. Related Works

In this section, we discuss the research works in main related areas and we highlight the contribution of this work in comparison with the reported works.

A. Experimentations in Cloud/DCs resource management and orchestration

A number of research works have been devoted to experimentations deployed in the DC environments to provide novel solutions addressing stringent requirements of communication intensive applications and user expectations. [12] [13] explore the use of SDN to improve data transfer rates or quickly recover from failures in DC networks exploiting programmability and flexibility features of OpenFlow. [14] analyzes historical data collected from a set of servers and focuses on optimizing VMs placement, while considering capacity constraints at these servers. In [15] authors present a framework for VM management that optimizes the placement of VMs, while requests arrive. The framework leverages applications topology as well as availability constraints also based on network monitoring data to avoid performance degradations at VMs due to network failures. However, these works do not address joint orchestration (i.e., allocation) of network and cloud resources thereby improving overall resource usage. Most importantly, in all these works the authors have considered small-scale laboratory testbeds to validate the proposed solutions while they had to resort to simulations in order to significantly show the merits of their proposals.

Other works in literature discuss the use of SDN-based solutions for resource orchestration in DCs and in some cases deployed them successfully [16][17]. In [18] the authors propose optimal algorithm for provisioning VMs and network bandwidth in a cloud computing environment, enabled by SDN capabilities and combining planned (i.e., in advance) and on-demand allocations. Due to limited resources, authors experiment the proposed solution in a small testbed. In [19] authors propose a platform for Software-Defined Clouds which allows for VM placements by jointly considering computing and network resource availability. The platform is then evaluated on an experimental testbed. These works consider instantaneous values of VM traffic to make allocation decisions. In [5] authors present an SDN-based orchestration prototype that not only controls the network, but also VMs and hypervisors to place VMs in optimized way. As for the network load, this

work considers average values of throughput exchanged between VMs over a certain time window. With respect to these works, our experiments demonstrate an orchestration process based on traffic load estimations resulting from a weighted average of a set of collected traffic data throughput values thereby making more effective data delivery path decisions and offering better than best-effort quality of service. Again, in both cases the proposals have a flaw on the scalability since their experiments have been carried out in a small-scale laboratory testbed.

ESCAPE [20] is an orchestrator prototype that allows for the coordinated deployment of cloud resources (i.e., VMs running network appliances) within DCs while connecting them dynamically through SDN. The validation of the prototype has been provided using an emulated environment (i.e., Mininet) and using a limited number of VMs and simple connections due to the very small size of the testing set-up, e.g., one single software SDN switch emulating an SDN interconnection network. Some advanced tests as regards the system architecture were presented in [21][22] where orchestration was performed across an abstract view of a multi-DC environment to address service chaining use cases and, thus, not considering resource usage details and dynamics at intra-DC level, that might be relevant aspects for DC/cloud operators.

B. Experimentations in Virtual Testbeds

Virtual testbeds offer open and large-scale virtualized infrastructures for researchers to perform experimentations in a Future Internet applications and network innovations. Thanks to ad-hoc experimentation frameworks, virtual testbeds allow to (i) combine special and different kinds of resources (robots, wireless, cloud), (ii) scale-up capabilities, and (iii) benefit from a number of tools to automate virtual testbed set-up and experiment operations (e.g., resource discovery and reservation, experiment control, measurements).

Many initiatives have been launched for the deployment of testbed federations. The most considerable initiatives in this direction are FIRE (Future Internet Research and Experimentations) in Europe [9], GENI (Global Environment for Network Innovations) [23] and SAVI (Smart Applications on Virtual Infrastructure) [24] [25] in North America. GENI provides a virtual laboratory for at-scale networking and distributed systems research and education, thereby promoting innovations in network science, security, services and applications. SAVI is a research testbed with the purpose of creating and delivering Future Internet applications. It allows for the deployment of SDN infrastructures within a virtualized environment composed of heterogeneous resources (e.g., FPGA, Software-Defined Radio, servers). FIRE offers cutting-edge testbed facilities aiming at enhancing the knowledge in Future Internet technologies through experimentations within multi-disciplinary and multi-technology testing environments (e.g., wireless and wire networks, cloud computing, automotive). Within these initiatives, a wide range of experiments have been enabled in many different fields, going from the evaluation of peer-to-peer mobile communications [26], Earth Observation emulation [27], the enhancement of medical applications with Future Internet capacities [28] and indoor localization estimation of robotic mobile nodes [29]. Moreover, within these initiatives international/European projects have funded to set-up testing facilities fostering experimentations in specific fields, such as SDN (e.g., OFELIA [6]) and Next Generation Internet (e.g., Fed4Fire [10]). Using these facilities, experiments have been carried out using SDN and cloud testbeds, e.g., [30]. However, these works did not tackle SDN-based orchestration as this work does. Moreover, as far as our knowledge, with respect to the works cited above, our experiment run on a larger scale. In fact, regarding the Fed4FIRE testbeds, on average requested slices include 5.3 nodes while our experiment required 38 nodes [31].

Public cloud platforms also offer the opportunity to benefit of virtualized capabilities and of a variety of services, including data analytics, machine learning, Internet of Things set-up [32][33]. However, the goals and intents they pursue are different, along with the users they target compared to FIRE or GENI. Indeed, they offer solutions especially to enterprises in order to help their business to scale and grow, foster more secure operations or enable cost savings. Moreover, despite the variety of services, some of cutting-edge technologies are not offered, such as 5G, as FIRE actually offers. Finally, services are not free and require payments to be issued which constitutes, for research activities, a real limitation.

C. Experimentations in Emulated Testbeds

A number of initiatives have been devoted to provide quite realistic emulated environments to perform experimentations, especially in the networking field. DOT [34] is a low cost and scalable network emulator, which provides guaranteed compute and network resources for the emulated components (i.e., switches, hosts and links). It can easily scale with network size and traffic volume, allowing to emulate large datacenters and wide-area networks. However, it is less realistic than Fed4FIRE regarding the cloud platforms. In fact, hosts are emulated by deploying user supplied VMs while in our case we could use physical machines, although remotely, thanks to bare metal services provided by Fed4FIRE platform. This way we could perform real lifecycle management operations to VMs (i.e., VM clone, run, shutdown, remove operations) after the installation of hypervisors and VM managers. Moreover, while the network embedding problem is trivial in the case of the Fed4FIRE testbed (the mapping of the tested network onto the physical infrastructure only depends on the number of available physical machines), in the DOT case it is more complicated and necessitates the use of a heuristic algorithm to minimize the translation overhead. Mininet-HiFi [35] is another network emulator that guarantees resource isolation and provisioning. It also monitors performance fidelity to help verify that an experiment is operating realistically. However, the performance of the network emulated on Mininet-HiFi still depends on the characteristics of the physical machine on which it is created. In fact, aggregate resource requirements of the emulated network must fit within the available single server. MaxiNet [36] is also a distributed emulator for SDN networks which spans the emulated networks on several

physical machines, thus allowing the emulation of large DC networks. Again, the performance of the emulator depends on the characteristics of the physical machines (even if more than one is used), which restricts its usage in heterogeneous environments. In fact, the load ends up to be evenly distributed over all physical machines, so the weak ones will become the bottleneck of the emulation. On the contrary, in the Fed4FIRE testbed, since bare metal services are supported, physical machines are made available to the experimenters and all have the same characteristics.

III. Virtual Testbeds and Orchestration Experiment Results

In this section, we describe the virtual testbed we set-up on top of the Fed4FIRE facility to perform experiments on SDN-based orchestration in Cloud DCs and the main components of the SDN-DC orchestrator prototype. We then report the experimental results we collected using the virtual testbed from the Fed4FIRE federated facility.

A. Fed4FIRE facility

Funded by the EU in the framework of Future Internet Research and Experimentation (FIRE) initiative, Fed4FIRE offers a heterogeneous, scalable and federated experimental facility in which a large number of European testbeds are integrated to provide a comprehensive testing environment covering various technology domains, including but not limited to, cloud computing, wireless and wired networks, sensor networks, and software defined networks.

A virtual testbed on top of the Fed4FIRE facility can be fully remotely operated by experimenters thanks to a comprehensive software framework offering a set of tools for the management of the overall experiment lifecycle (e.g., tools for resource discovery and reservation, experiment control, measurements) and key features of trustworthiness (e.g., federated identity management and access control, accountability, SLA management). Hence, the experimenters can securely access all the required resources with a single account and focus on his/her core task of experimentation instead of dealing with specific practical aspects of each testbed (e.g., request of different accounts, learning different technicalities).

Regarding the access to the testbeds, a set of tools allow experimenters to specify and allocate a slice of virtual resources (i.e., virtual servers, VMs and virtual links) to carry out his/her experiments (i.e., *experiment container*). More specifically, the MySlice Web portal offers an ingress point to experimenters to access the Fed4FIRE federation and its facilities, while the jFed platform provides additional tools to check the resources availability and, eventually, reserve resources in the federated testbeds to run automated tests, and to provision and manage experiments [37] [38]. The experimenters can use the *experiment container* to host a set of VM images running the software modules of his/her system under test and start performing experiments. The experimenters have also the possibility to work with ready-to-use VMs selected from a toolkit with a large set of available VM images equipped with auxiliary services (e.g., Operating Systems such as Debian, Ubuntu, CentOS) and software appliances (e.g., Open vSwitch (OVS) [39], POX [40]). Finally, experimenters can benefit of a set of tools (e.g., Zabbix, Nagios, Collectd) for monitoring and measurement tasks during the tests.

Regarding the computation and communication efficiency of the cloud services running on the allocated resources, Fed4FIRE guarantees that the performance experienced by a user is in line with the capabilities required during the resources reservation phase and is not affected by other users. More specifically, Fed4FIRE is not a public virtual environment, but a real testbed that makes available physical machines (i.e., bare metal machines) where perfect isolation from other experiments running on the same testbed is provided at the computing level. At the network level, the links capacity is also guaranteed through gigabit ethernet links, partitioned among vlans in the switches. In such a case, the background load has no effects on the obtained results and the required throughput is guaranteed (no packets delay/loss is experienced if not explicitly configured by the experimenters) [41]. Finally, the facilities offered by Fed4FIRE are for free while the access to the most interesting services of public cloud platforms such as AWS [32] and GCP [33] is subject to charges, which constitutes, for research activities, a real limitation.

B. Virtual DC Testbed and SDN-based Orchestration Set-Up in Fed4FIRE

The virtual testbed we set-up on Fed4FIRE for the experiments on cloud and SDN orchestration reproduces a DC environment with a set of components devoted to orchestration and resource control tasks. To this purpose, we established two experiment containers within the Virtual Wall testbed [42] which was identified as the most appropriate platform for our experiments among the different testbeds provided by Fed4FIRE. Indeed, the Virtual Wall testbed offers a large number of VMs to experimenters thereby providing experiment containers with large capabilities as we required to set-up a DC testbed with a significant number of switches and servers. The Virtual Wall testbed also provided us with the capability of adjusting the network state (e.g., traffic load) according to the experiment needs. Moreover, it allowed for setting any kind of network impairment (i.e., latency, packet loss) through specific tools (e.g., iperf) thereby making experimental scenarios much more realistic. Finally, it gave us the possibility to use ad-hoc scripts to automate a set of testing operations, e.g., monitoring data collection. Fig. 1 shows the two experiment containers we set-up, each of the them being composed of a pool of VMs that we used to deploy the needed software components.

The first experiment container (left) is composed of 37 VMs that we used to reproduce the overall DC environment. More specifically, we used 20 VMs to deploy OpenFlow soft switches by means of OVS [39]. Then, the OVS switches were interconnected through virtual links according to a fat-tree topology with 3 switching layers, i.e., edge, aggregation and core. Other 16 VMs were used to deploy virtual servers. Indeed, those VMs were equipped with the Xen Cloud Platform (XCP) [43] which is a Virtual Machine Manager (VMM) also providing the Xen API tool stack that supports lifecycle

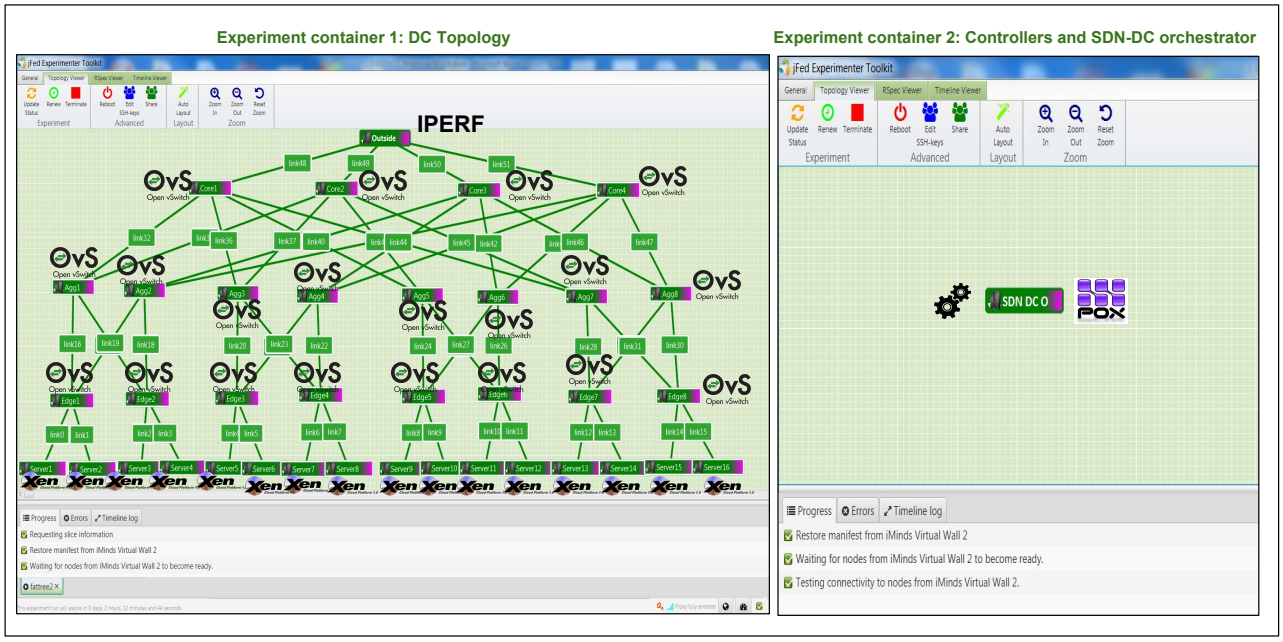

Fig. 1. Virtual Testbed set-up: Experiment containers.

management capabilities for the VMs running in the virtual servers (i.e., VM clone, run, shutdown, remove operations). The remaining VM was used to run an outside host that acted as a sink for the traffic generated by the VMs installed in the virtual servers. More specifically, we used the iperf tool [44] to generate Variable Bit Rate (VBR) traffic from the running VMs to the outside host. The second experiment container (right) is composed of one VM (i.e., orchestration node) and is used to deploy the orchestration components which consist of the SDN-DC orchestrator along with an SDN/OpenFlow controller (i.e., POX controller [40]) managing the OVSs in the other experiment container over a secure connection using the OpenFlow protocol.

The main blocks of the SDN-DC orchestrator are shown in Fig. 2. It consists of a *Resource Orchestration Engine* (i) exploiting APIs offered by the POX controller and the XCP running in the servers to control and manage resources (i.e., switches and servers, respectively), and (ii) elaborating requests for VM set-up and termination coming from the VM Request Manager. Upon the arrival of a VM set-up request from the VM Request Manager, the Resource Manager extracts the VM specifications (i.e., number of CPU cores required for a proper data processing rate and the bandwidth demand to sustain the VM traffic). Such specifications are then used to perform the combined selection of (i) one server able to host the VM, and (ii) a sequence of switches able to sustain the required VM traffic load in the DC network. To perform the selection, the *Resource Manager* leverages a Selection Algorithm that uses load status data related to both servers and switches that are stored in the Host Information Database and Network Information Database, respectively. According to the selections, the *Resource Manager* triggers the coordinated provision of (i) the VM at the selected server, and (ii) the data delivery path across the selected switches by leveraging

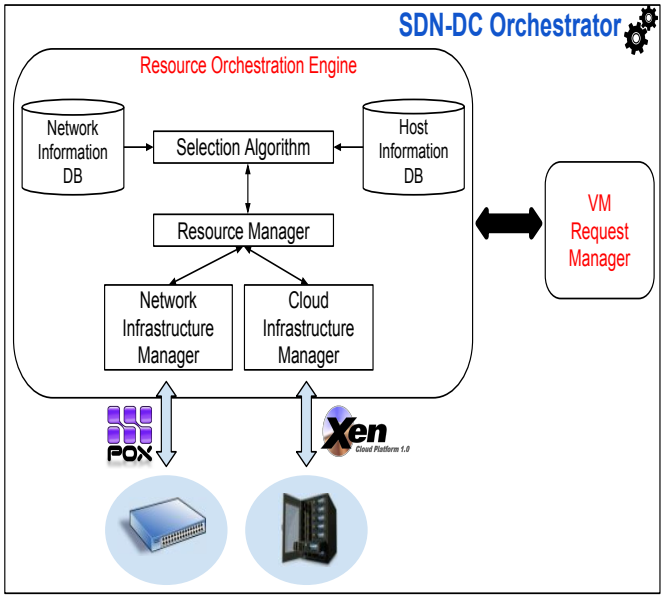

Fig. 2. Software Components of SDN-DC orchestrator.

technology-independent APIs offered by the *Cloud Infrastructure Manager* and the *Network Infrastructure Manager*, respectively. This way, underlying either technology-specific resource controllers can be used. In this case, the *Network Infrastructure Manager* leverages the POX OpenFlow Controller to enforce forwarding instructions to selected switches through OpenFlow messages, and the *Cloud Infrastructure Manager* leverages the XCP to set-up and stop the VMs into the servers.

The process of server and switch selection is a multi-step process that is constrained by the multi-layer DC topology, where the selection of the server restricts the possible

sequences of switches that can be chosen and vice versa. Firstly, two different orders can be followed to select the server and the switches. Correspondingly, we conceived two categories of algorithms: (i) Server-Driven (SD) that first attempts to find one server for the VM and then a sequence of three network switches/links for the data delivery path across core, aggregation and edge layers; (ii) Network-Driven (ND) that first attempts to find a sequence of three network switches/links for the data delivery path across core, aggregation and edge layers and then a server where to deploy the VM. At each step of server and switch selection, multiple choices are possible as result of a redundant tree-like DC network topology and multiple servers connected to each edge switch. Correspondingly, we considered two bin-packing policies to select the switch/link and server at each step of selection: (i) First Fit (FF) that allocates the VM (traffic load) in the first-indexed server (switch) provided that the required CPU cores (bandwidth) are (is) available to meet the request; (ii) Worst Fit (WF) that searches for the most unloaded server (switch) to allocate the requested VM (traffic load), i.e., where the largest amount of CPU cores (bandwidth) is available provided that it is sufficient to meet the request. Ultimately, four possible selection algorithms can be adopted in the SDN-DC orchestrator by combining the two resource selection orders with the two bin-packing policies, i.e., SD-FF, SD-WF, ND-FF and ND-WF. For the sake of clarity, these algorithms are summarized in Table. I

TABLE I
SELECTION ALGORITHMS DESCRIPTION.

Algorithm	Description
SD-FF	Server-Driven First Fit
SD-WF	Server-Driven Worst Fit
ND-FF	Network-Driven First Fit
ND-WF	Network-Driven Worst Fit

Whatever algorithm is used, the selection of server and switches is performed based on load estimations carried out by the Resource Manager and made available to the Selection Algorithm. As for the server load, the number of CPU cores already allocated to run VMs retrieved from the XCP are used. As for the network switch load, OpenFlow traffic statistics are exploited that are available from the POX Controller. More precisely, values of per-port received/transmitted bytes counters of each switch are used that are queried periodically through a daemon that triggers the issuing of OpenFlow commands to switches through the POX Controller. Then, the daemon computes the delta of received/transmitted bytes between two consecutive queries, and consequently the traffic rate at the switch ports over time by dividing the resulting value by the duration of the polling time [45]. Based on this information, the Resource Manager can build consolidated trends of traffic load at switch ports by making averages of traffic rates over time according to an exponential weighted moving average scheme [8]. Basically, the *Resource Manager* computes the weighted average between the last computed (i.e., instantaneous) value of the traffic rate at a specific switch port and the previously obtained (i.e., historical) average values. The result of such an averaging process can be the smoothing of short-term fluctuations and the highlighting of longer-term trends or cycles in the traffic rates at the switch ports thereby the influence of past values decreases exponentially. Accordingly, the *Selection Algorithm* is able to track the switch load while making selection decisions with a level of reactivity that depends on the relative weight given to the instantaneous and historical values (i.e., history weight α) in the average process. In fact, giving weight to instantaneous values allows for responding to fluctuations in a faster way, whereas giving more weight to the historical values allows for smoothing out short-term load fluctuations and highlighting longer-term trends or cycles. Also the number of historical values (i.e., size of the monitoring window M) considered in the averaging process is significant to highlight specific trends in the switches/links utilization. Finally, the interval used to query the switches and to compute traffic rates (i.e., polling interval ΔT) is relevant. Indeed, having a long polling interval might miss the bursty flows whereas a short polling interval could increase the processing load at the SDN orchestrator and worsen the performance.

C. SDN-DC Orchestrator Performance Evaluation

In this section, we report the experimental results collected from the testing campaign to evaluate the performance of the SDN-DC orchestrator using the virtual DC testbed described in the previous sections.

Table. II summarizes the settings we used to perform the experiments. We generated the requests for VM allocations (i.e., VM requests) according to a Poisson process characterized by inter-arrival (IAT) and holding times exponentially distributed with an average of $1/\lambda$ and $1/\mu$, respectively. $1/\lambda$ varies within the range $[80s, 200s]$, whereas $1/\mu$ is fixed to 10.000 seconds. Each request includes the demand for (i) computational power expressed in number of CPU cores and uniformly distributed within the interval $[2, 4]$, (ii) bandwidth (i.e., traffic rate) expressed in Mb/s and uniformly distributed within the range $[20, 80]$, (iii) storage capacity equal to $1GB$, and (iv) RAM also equal to $1GB$. Regarding the DC initial configuration, the computational power of the servers is fixed to 20 CPU whereas the links capacity is fixed to $80Mb/s$. Results are plotted for each resource selection algorithm (i.e., SD-FF, SD-WF, ND-FF and ND-WF) and, if not otherwise specified, considering ΔT equal to $35s$, the history weight α equal to 0.2 and a monitoring window M equal to 6 values.

To load the network links, we have created a script that generates variable data streams from the allocated VMs towards the outside host following the selected paths. More specifically, we used the *iperf* tool to send variable TCP traffic in line with the traffic demand in VM request. We first started an iperf process in a server mode as a traffic receiver at the outside host, then at every allocated VM, another iperf process was started in client mode as the traffic sender for the whole duration of the holding time.

To evaluate the orchestrator performance, we consider as a baseline the case of a random selection of the server and the use of the spanning tree algorithm to select the path followed by VM traffic data. This baseline reproduces a common practice in DCs where neither the resource management operations

TABLE II
EXPERIMENT PARAMETERS.

Parameters	Values
Average inter-arrival time $1/\lambda$	$[80s, 200s]$
Average holding time $1/\mu$	$10.000s$
VM computational power	$[2, 4]$ CPU cores
VM bandwidth	$[20Mb/s, 80Mb/s]$
VM storage capacity	$1GB$
VM RAM	$1GB$
Number of servers	16
Number of switches	20
Servers computational power	20 CPU cores
Link capacity	$80Mb/s$
α - History weight	0.2
ΔT - Polling interval	$35s$
M - Size of the monitoring window	6

(e.g., server selection) nor path selections are engineered [46]. Indeed, as for the path selection, load-balancing approaches are used, such as the Equal-Cost Multi-Path (ECMP), which under-utilizes the network links by wasting on average 61% of bisection bandwidth [47] [46]. By adopting the spanning tree algorithm in the baseline we emulate the same behavior by not using the redundant links, which can be considered an acceptable approximation. Finally, for a sake of fairness in the comparison with orchestration algorithms, the admission control based on the load on the network links is carried out also in the baseline.

In the following subsections, the experimental results are reported divided by category of performance indicators and performance analysis.

1) Blocking Probability analysis: We assessed the Blocking Probability as the percentage of requests that are rejected due to the lack of resources (i.e., either network bandwidth or servers CPU cores) to satisfy the VM allocation request. The obtained results are plotted in Fig. 3 as a function of the IAT. As expected, the blocking probability decreases as the IAT values increase because in general more resources are available as the VM request is processed. Moreover, all the orchestration algorithms significantly outperform the baseline which does not utilize all the redundant links at the aggregation and core levels, thus de-facto reducing the available network resources and increasing the blocking probability. More specifically, the ND algorithms present the lowest rejection rate and a gain of around 5% is achieved with respect to the SD algorithms as the network resources are more efficiently used in the ND case [7].

2) Analysis of Resource Utilization rates: We assessed the Resource Utilization rate as average switches load and percentage of active servers out of the total number of servers. For this analysis and the following ones we present results for an IAT fixed to $150s$.

Fig. 4 plots the switches load for the orchestration policies. As expected, the baseline presents the lowest average since 50% of the available links are not used which lowers the amount of traffic transiting over the switches and then decreases their average load. Regarding the other policies, although the values are very close, we notice that in the SD-FF case the average is slightly lower which can be explained as follows. The values of this figure correspond to the case where the IAT is equal to $150s$. As previously shown in Fig. 3, the blocking probability is higher in SD-FF than in the other policies, which means that less VMs are allocated and then lower traffic is injected into the network. Such reduction in the amount of traffic is reflected in a decrease of the overall switches load. However, the peculiarities of the DC network topology with three fat-tree levels and with over-subscription adopted at the core levels, calls for an in-depth analysis of the switch load.

In Fig. 5 we split the average switches load according to the three fat-tree levels, i.e., core, aggregation and edge levels. We can observe that the baseline presents the highest average load at the core level (around 15% on average more with respect to our algorithms) which is explained by the actual use of only one core switch that handles all the traffic, due to the underutilization of resources caused by the application of the spanning tree algorithm. Regarding our resource selection algorithms, we notice that the average load at the core level is still higher than at the other network levels with almost the same at the aggregation and edge levels. This is mainly due to the over-subscription ratio of $1:2$ illustrated by a number of available switches at these two levels higher than at the core level, which burdens the core switches while reduces the average load at the aggregation and edge switches.

Going into more detail on the distribution of the load among switches, in Table. III and IV we present the load at the Edge Switch 1 and Edge Switch 3, respectively (going from the left to the right in the established DC fat-tree topology).

TABLE III
LOAD OF EDGE SWITCH 1 [%].

Algorithms	Peak Value [Mb/s]	Mean Value [Mb/s]	Standard Deviation [Mb/s]
SD-FF	70.99	43.66	17.93
SD-WF	54.70	24.62	13.64
ND-FF	41.06	19.33	15.52
ND-WF	38.91	21.21	14.06
Baseline	66.02	32.51	17.92

TABLE IV
LOAD OF EDGE SWITCH 3 [%].

Algorithms	Peak Value [Mb/s]	Mean Value [Mb/s]	Standard Deviation [Mb/s]
SD-FF	55.75	31.89	11.87
SD-WF	40.90	18.44	12.004
ND-FF	32.12	18.54	7.5
ND-WF	37.27	21.77	11.56
Baseline	56.67	32.37	15.37

The obtained values show that Edge Switch 1 has the highest load peak values in case of SD-FF algorithm. In fact, the servers are chosen according to the order given by their id value, i.e., the server with the lower id is firstly selected if it has enough capacity, with a resulting consolidation of selections in the first-indexed servers. Since the servers are directly connected to the edge switches, the selection order of the Edge Switch follows the same order of server selections, also resulting in a consolidation of load in the first-indexed Edge Switches. Indeed, the same trend can be observed for

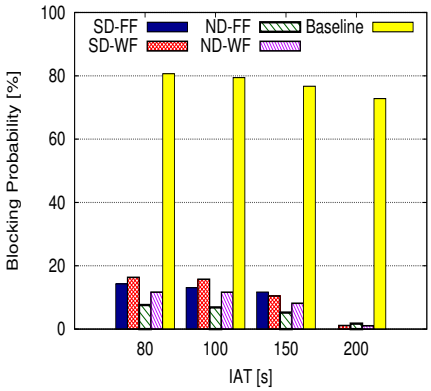

Fig. 3. Blocking probability vs. IAT

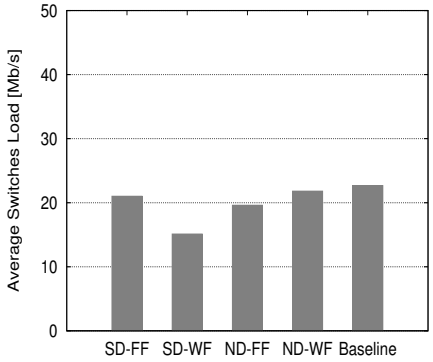

Fig. 4. Average switches load

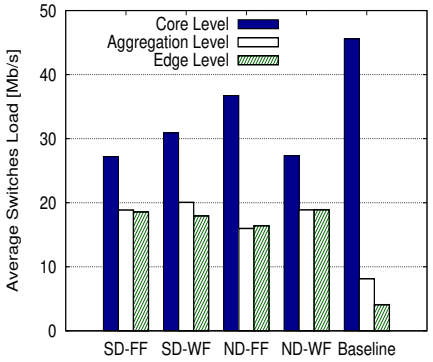

Fig. 5. Average switches load Fat-tree levels

the Edge Switch 3. Also in the baseline, the Edge Switch 1 and Edge Switch 3 are significantly loaded. This is instead due to the use of a more restricted set of resources with respect to the orchestration algorithms (i.e., redundant links not used) which allocates the load on a subset of switches. Also due to the random selection of servers, this results in a similar average utilization of Edge Switches as shown in Table. III and IV. On the other hand, ND orchestration algorithms present the lowest mean and peak load values. In fact, since the selection of the path is performed starting from the core network level down to the edge level (i.e., the core switch is firstly selected, then the aggregation switch and at the end the edge switch), the choice of the path is biased and depends mainly on the selections in the previous network levels. This results in a spreading of utilization of switches, and thus of edge switches and servers, even in the case of ND-FF despite the intrinsic consolidation effect of FF policy. Because of the limited space, and without lack of generality, we restricted the measurements to only two switches.

Finally, in Fig. 6 we plot the percentage of active servers for the four algorithms with respect to the baseline, at half and at the end of the experiment. By active servers we mean the servers that contain at least one running VM when the results are collected. Results show that the FF algorithms present a lower number of active severs at half experiment since they tend to consolidate the allocations on a subset of the resources (either servers or switches/links). Moreover, due to the spreading of allocations in WF algorithms, 87% and 94% of the servers are on at half experiment in the ND and SD case, respectively. The same trend is observed at the end of the experiment where all the servers are on in the WF case while only around 50% of the servers are active in the FF case. For the baseline the number of active servers remains always the same mainly because of the limited set of used links/switches (i.e., in the spanning tree algorithm, the redundant links are not used), which makes the algorithm choose almost always the same subset of servers.

3) Data throughput and latency analysis: The effectiveness of the SDN-DC orchestrator is demonstrated by the higher acceptance rate of VM requests coupled with a significant improvement in terms of resource utilization. However, this can be obtained at the cost of an increased risk of data throughput degradations at switches. Indeed, the traffic generated by the allocated VMs is subject to high fluctuations in Cloud DCs, and correspondingly the switches might be overloaded at a certain time and underloaded few seconds later or vice versa. Data throughput degradations are minimized in our work since the admission decision and the selection of switches are made by the SDN-DC orchestrator based on always-on monitoring data collection about switch port usage to estimate trends in the switches load and to make an educated guess of the capability of switches to sustain the required traffic rate (i.e., bandwidth demand). However, as an estimation process, it is affected by an intrinsic although minimal risk to incur in switches overload cases. This is especially true for core switches that in the fat-tree DC network are subject to over-subscription. In case of overload, data throughput at switches could be degraded with impact in terms of latency experienced by data packets.

For this reason, we considered as a performance indicator the Degradation Index (DI) accounted as the percentage of time the links are overloaded (i.e., reaching 100% of utilization) which introduces delays in the delivery of the packets. In fact, during the tests, since the VMs holding time is very high ($10.000s$ on average), the load of the network should continuously increase and so the utilization of the links. However, due to the high variability of the traffic, such effect is mitigated and the actual status of the network fluctuates constantly. During the tests we collect the statistics every ΔT seconds, and, then at the end of the experiment, we calculate DI which accounts for the number of times the actual links utilization reached 100% over the total number of statistics occurrences. As result, the DI takes into account the likelihood of degradations in the data delivery that might be experienced by users.

In Fig. 7 and Fig. 8 we plot the average DI for the core switches and the DI per each core switch, respectively, with IAT fixed to $150s$. Results show that FF algorithms tend to have higher DI values as expected due to the resulting consolidated selection of resources that increases the usage of a restricted set of links/switches. Indeed, during the tests we noticed that especially in the ND-FF case more links have a load within the $[80\%, 100\%]$ of link capacity. This usage profile definitely increases the risk of overload for those links/switches and thus of data throughput degradations in the

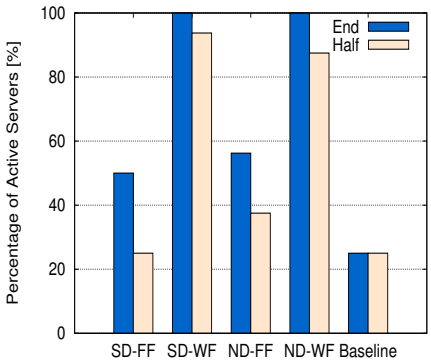

Fig. 6. Percentage of active servers

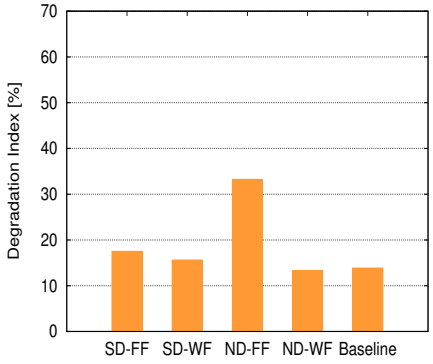

Fig. 7. Average Degradation Index for core switches

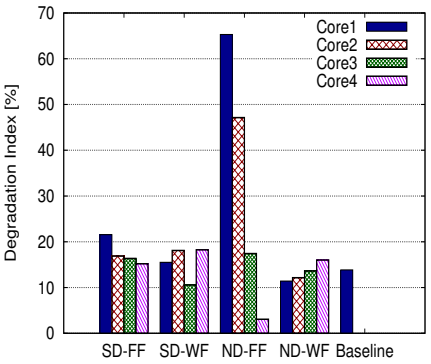

Fig. 8. Degradation Index per core switch

ND-FF algorithms. Moreover, this is in line with the average switch load in FF cases which was around 20% (as shown in Fig. 4) since the consolidated usage of resources is masked by the average process. While the average DI in the SD-FF is higher than in the other algorithms, the difference is significantly lower in the ND-FF case, since the selection of the links is biased by the selection of the server which slightly scatters the link usage across the network thus decreasing the link overload occurrence. The WF algorithms present a lower DI because of the intrinsic scattering of selections performed both at the server and switch/link level that significantly alleviate the occurrence of data throughput degradations. The fact that all the algorithms present DI values higher than the baseline is to be considered as a cost to pay for having higher request acceptance and resource utilization rates (and thus revenues).

The DI results are in line with the measurements on the delay (i.e., latency) experienced by packets across the network. By delay we refer to the Round Trip Time (RTT) which is the time required for a packet to travel from a specific source to a specific destination and then get back again. Once a VM is allocated on a server, we use the ping utility installed on the outside host to start measuring the RTT from the outside host to the VM for the whole duration of the holding time. Table. V summarizes the carried out measurements. For sake of brevity, we show the ones taken for only one server (i.e., the RTT values for a ping between a VM allocated on server 3 and the outside host). Also in these results, since we are interested in studying the occurrence of link overload impacting the latency performance, mean values might be not significant. For this reason, we will comment on the peak values. We can see that the ND-FF presents the highest latency peak value and then the lowest data throughput experienced by packets. In fact, as long as enough bandwidth is available, the ND-FF algorithm consolidates link selections, thus tending to congest a subset of them. Among the other algorithms, ND-WF presents the lowest peak value since it distributes the load on all the available links. The SD algorithms present almost the same results since the choice of the network path is performed after the selection of the server which then limits the choice. With respect to our algorithms, the baseline has a lower peak value but presents a higher average. In fact, despite the high variability of the traffic, the limited set of available links at the aggregation and core levels minimizes the resources selection possibilities which limits the latency fluctuations but increases its average. Finally, the WF algorithms present a lower standard deviation with respect to FF algorithms and baseline, as result of a more uniform resource usage pattern.

TABLE V
NETWORK LATENCY.

Algorithms	Peak Value [ms]	Mean Value [ms]	Standard Deviation [ms]
SD-FF	0.986	0.589	0.108
SD-WF	0.985	0.555	0.085
ND-FF	1.04	0.584	0.107
ND-WF	0.847	0.597	0.086
Baseline	0.706	0.641	0.0207

TABLE VI
SERVICE SETUP TIME: MEAN VALUES AND CONFIDENCE INTERVAL AT 95%.

Algorithms	Algorithm Computation time [ms]	Confidence Interval	Total Setup time [ms]	Confidence Interval
SD-FF	36,85	5.6	42525,96	201,37
SD-WF	38,21	8.7	38265,97	646,18
ND-FF	15,96	5.84	40396,4	957,48
ND-WF	17,77	4.46	38813,01	721,42
Baseline	21,47	6.25	38122,28	872,59

4) Setup time analysis: We assessed the set-up time computed as the time needed to fulfill a VM allocation request. Table. VI reports the set-up time split into the resource selection time and the total request processing time (both reported with the mean values and the confidence intervals at 95% and for IAT fixed to $150s$.). In addition to the server and path selection time, the processing time includes the VM cloning time, the IP address assignment time by the DHCP server and the time needed for the setup of the forwarding rules along the selected path. The server and path selection time is higher for the SD algorithms and the baseline, since the number of servers to check is higher than the number of switches. Specifically, the baseline has a lower selection time, since after the choice of the server the path is already pre-

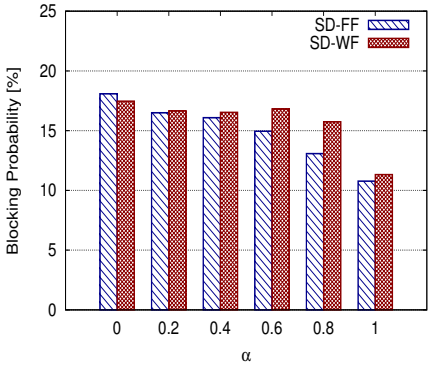

Fig. 9. Impact of α on the Blocking probability

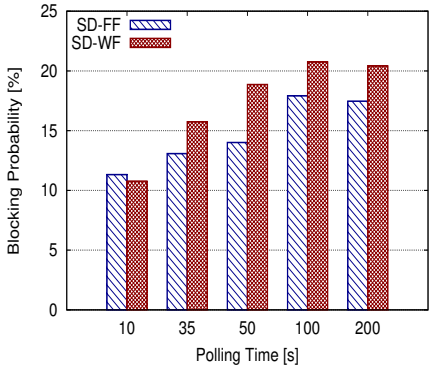

Fig. 10. Impact of ΔT on the blocking probability

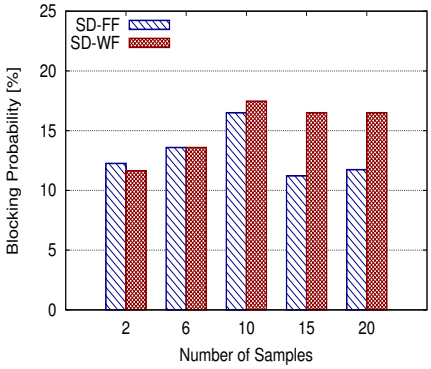

Fig. 11. Impact of M on the Blocking probability

configured. Regarding the two SD policies, the WF takes more time since it checks all the available resources before selecting the most unloaded ones, while the FF policy takes the first available server/switch. Finally, the overall processing time is equivalent for all the algorithms and almost independent from the network load. In fact, the main difference resides in the resources selection time which is however negligible (tens of milliseconds) with respect to the VM cloning time (around 4 seconds) and the DHCP time (around 35 seconds) that are almost the same for all the algorithms.

5) Analysis of the impact of the estimation parameters: We assessed the impact of the load estimation parameters (i.e., α, ΔT and M) on the SDN-DC orchestrator performance to derive design guidelines helpful to properly tune the estimation process while reducing the processing burden and the risk of data throughput degradations. For sake of brevity, we report the results for the tests related to the SD-FF and SD-WF algorithms.

Fig. 9 plots the blocking probability as a function of the parameter α. Results show that by increasing α, the blocking probability slightly decreases. In fact, a higher α corresponds to giving more weight to the instantaneous values of the traffic leading to estimation values changing as rapidly as current load values thus with minor filtering of spikes [8]. This leads to a beneficial effect in terms of blocking probability since this variability of estimations presents more available resources useful for allocating new VMs. However, as shown in Fig. 12 this lower blocking probability is paid in terms of higher degradation index since accepting more VM requests leads to higher likelihood of data throughput degradation. In case of SD-FF algorithm the values of blocking probability are lower (and correspondingly the DI values higher) than in SD-WF since the consolidation of allocations across servers and links/switches (instead of spreading of allocations) leads to multiple releases that are more likely to concern the same links/servers which increases the probability to have available resources that meet the requests requirements thus improving the rejection rate.

Fig. 10 presents the effect of ΔT on the blocking probability. We notice that by enlarging ΔT, the blocking probability slightly increases. In fact, a larger statistic polling interval means using a lower number of measurements and thus less precise estimations. Overall, this leads to increasing the rejection rate because the orchestrator is unaware about the VMs that were released and the traffic variability within the interval of time separating two consecutive statistic collections. The increase of the blocking probability is slightly lower in the case of SD-FF algorithm which takes advantage from the high variability of the traffic to condensate more VMs on the servers and thus more traffic in low-indexed switches/links. However, as shown in Fig. 13, with less requests that are admitted, lower DI values can be observed as ΔT values increase.

Finally, Fig. 11 plots the blocking probability as a function of the number of samples M in the monitoring window. We can observe that by increasing M the blocking probability slightly increases since the historical traffic load values are more considered thereby highlighting more long-term trends while smoothing load variations in the short term. As result, the orchestrator takes less advantage from the variability of the traffic to allocate new VMs and the blocking probability increases. However, such effect is restrained and we can see that above a certain limit (10 samples), the blocking probability remains stable in case of the SD-WF algorithm whereas it slightly decreases in the SD-FF case.

The fluctuating trend in both the SD-FF and SD-WF cases can be explained by the impact of the value of α on the plotted results. In fact, in Fig. 11 α is equal to 0.2, which, as previously explained, gives more weight to the historical samples and then leads to decrease the allocation of new VMs when M is low. On the contrary, adopting higher M values introduces more historical values which then confirms the trend of a higher Blocking Probability as already explained in Fig. 9. As shown in Fig. 14, the DI values decrease as a result of adopting higher M values thus confirming the better estimation of the historical values of the traffic and the lower effect of the traffic variability on the system performance. Moreover, as previously explained, DI is slightly lower in the SD-WF algorithm thanks to its scattering effect.

IV. Discussion on Virtual Testbeds and Comparison

In this section we analyze the advantages offered by virtual testbeds compared to other testing facilities in terms of effectiveness in experimentations. To deepen this analysis, we fur-

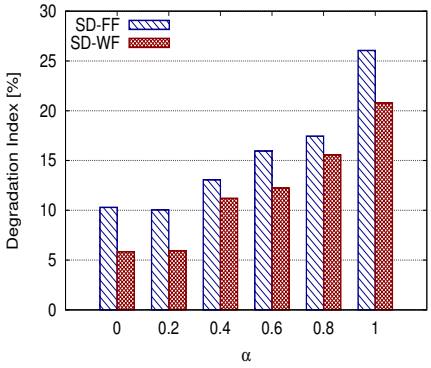

Fig. 12. Impact of α on the Degradation Index

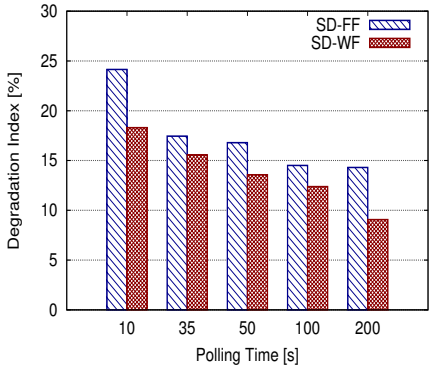

Fig. 13. Impact of ΔT on the Degradation Index

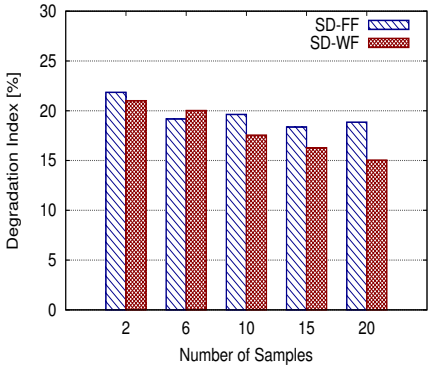

Fig. 14. Impact of M on the Degradation Index

ther discuss the experimental results presented above against results obtained while testing the SDN-DC orchestrator in laboratory [8] [48] and simulation testing environments [7] [49].

A. Virtual Testbeds: advantages and comparison with Laboratory Testbeds and Simulation Environments

Virtual testbeds provide relevant benefits that make them very attractive for experimentation.

First, the experimenter can evaluate the performance of the system under test without the need of owning and operating an experimental facility. Indeed, the set-up of an experimental facility generally demands costly, cumbersome, and time-consuming efforts, especially in case of DC testbeds that are featured by a very large number of devices and links (i.e., switches, servers, cables). With virtual testbeds, the experimenter can start performing significant tests of the system prototypes before passing them to the production phase, thereby achieving evaluation process effectiveness and cost savings.

Second, the use of VM images allows the experimenter to manage the testbed in a straightforward way, especially in case of DC experiments where a high number of identical devices (i.e., switches, servers) are deployed. Indeed, in case of a laboratory testbed, the management and configuration operations on identical devices could be highly time-consuming and error-prone. Instead, in virtual testbeds, the configuration operations are performed only once, while building the VM image. Then, the VM image is saved and used (i.e., through cloning) as many times as needed into the other systems, thus significantly simplifying and reducing setup times. Hence, the experimenter is relieved from spending plenty of time in case of crashes to completely reconfigure the testbed or in case of repetitive testing operations.

In Table. VII, we provide insights on the comparison of virtualized testbed environments with other well-known testing environments (i.e., laboratory testbeds, emulators, simulators) in terms of experiment scale, fidelity, usability of experimenter software modules, and experiments repeatability. As for laboratory testbed we refer to a set-up typically in university-owned facility for conducting experiments and testing of theories, tools and, in general, new technologies.

Virtual testbeds generally offer high scales through large testbed facility and testbed federation. The scales offered by simulators (e.g., ns-2 [50]) are higher (theoretically unlimited), since they reproduce the operation of the system under test through a software program. On the other hand, the experimental scale for laboratory testbed is very limited, especially in the case of DC testbeds, due to the costs for purchasing and upgrading equipment. One possibility to slightly mitigate this scale constraints could be the use of virtual servers in the laboratory (e.g., local OpenStack deployment). However, such solution still present the same shortcomings because ultimately number of virtual servers are still constrained by the availability of physical servers or servers capacity in the laboratory. The emulators stay in-between by offering higher scales than laboratory testbeds, but still lower than simulators because the scale is dramatically constrained by processing capabilities of hosting physical machines in terms of CPU cores and RAM. For instance, Mininet [51] allows to reproduce the operation of a complete network, including hosts, links, and switches on a single host machine. However, network topologies generally need to be scaled down (e.g., dozens of servers/switches with low links capacities) to have the experiments running in expedited way.

TABLE VII
COMPARISON OF THE VALIDATION ENVIRONMENTS.

Testing Environment	**Scale of Experiment**	**Usability of experimenter software modules**	**Level of Realism**	**Level of Repeatability**
Virtual Testbed	High	Yes	High	Quite Low
Laboratory Testbed	Very Low	Yes	Very High	Low
Emulator	Low	Yes	Quite High	Quite Low
Simulator	Very High	No	Quite Low	Very High

The possibility to use software components developed by the experimenters in the virtual testbed, laboratory testbeds and emulators allows to carry out experiments directly using the same software modules as the real system. This possibility is not supported by simulators. This has an impact in terms of level of realism, i.e., fidelity, that can be achieved by the testing facility. Both virtual testbeds and laboratory testbeds

allow to reproduce real system capabilities with high fidelity through the usage of the same software modules as the real system, although running in an experiment container. The emulators offer quite realistic testing environments. For instance, Mininet's virtual hosts, switches, links, and controllers behave like real components although operated using software rather than hardware (e.g., software switch operations reproduced using OVS). On the other hand, the fidelity to the real system offered by simulators is generally quite low since they just synthetically reproduce (i.e., through a software program) the operation of the system under test. Ideally, if the simulation model was accurate, the reproduced system operation would be highly realistic. In practice, some shortcuts to the simulation model are likely to be adopted and thus often some operational aspects are just approximately reproduced for feasibility reasons (e.g., data plane modeled as program data structures instead of real flows of packets), with consequent impact on the realism of the obtained results.

Finally, in terms of level of experiment repeatability [52], simulators allow to perform a high number of experiment repetitions with the same set-up and system operation, which definitely improves the statistical significance and the accuracy of the results that can be obtained. In general, this is not true in the other testing facilities, including virtual testbeds, because it is not always possible to repeat the experiment in a short period of time, due to supervening resources unavailability or failures, software configuration or management issues.

B. Discussion of Orchestration Results against laboratory testbed and simulation results

The Blocking Probability results obtained using the virtualized testbed follow the same trend as simulations. This demonstrates that the testbed we set-up has both the appropriate scale and the proper level of realism thereby effectively reproducing the actual system capabilities. In simulations, we could effectively evaluate the blocking probability (i.e., with the system in a steady state conditions) since we could arrange high scales of resources and we could stress the system with a large range of request rates and service durations. The main drawback was that the traffic data was synthetically produced and we could not collect some performance indicators in terms of network load (e.g., occurrence of data delivery degradations). With respect to experiment results, we obtained, at comparable rates and resource demands, lower blocking probability values and a more significant difference among the orchestration algorithms thereby better highlighting algorithm peculiarities in resources utilization [8]. In laboratory testbeds, we had to struggle with resource constraints to reproduce DC environments especially in terms of network topology. This definitely resulted in higher loads for switches and servers thus producing not only higher request blocks but also hiding peculiarities in the resources utilization profiles in different algorithms and smoothing out differences in the blocking probability values. The opportunity to set-up the virtual testbed definitely allowed us to obtain the performance evaluations in a significant scale while using directly software modules under test and TCP traffic. Thanks to the monitoring features made available by the testing infrastructure, we could perform a more comprehensive assessment of the orchestration algorithms in terms of Resources Utilization rates, which was not possible during the simulations where the load was deduced from the nominal values of the allocated requests as result of the data plane synthetically reproduced. Moreover, we performed a more extended and sophisticated set of measurements in terms of number of evaluated switches/links while leveraging higher accurate monitoring data and in a more realistic scenario. Indeed we could reproduce a fat-tree network with a significant number of switches and connected servers with respect to laboratory experiments and we could use a realistic traffic in contrast to simulations. As for server utilization, with respect to our previous studies where we evaluated the number of VMs per server in the experimental tests and the overall servers utilization/occupancy in the simulations, in this paper we gave a more comprehensive evaluation considering the overall profile of servers utilization by assessing the total number of active servers. In fact, in the experiments, the number of servers was very low (6 with respect to 16 here) with low capacities (less than 10 VMs were able to be allocated on each server) which prevented us from performing a comprehensive assessment of VMs placement profiles. In the simulations, the synthetic data plane prevented us from doing fine-grained evaluations on server and switch capacity occupancy. On the other hand, the focus was mainly on the effectiveness of the orchestration process on a long run which we could assess by doing long-term evaluations with the system in steady status conditions (i.e., after the allocation/release of around 1000 VMs) which was really challenging to achieve using experimental facilities, either laboratory or emulated ones.

The Degradation Index results were overall in line with the ones obtained in the simulations in terms of different DI profiles we obtained for the orchestration algorithms, i.e., FF algorithms tend to have higher DI values than WF algorithms due to the consolidated usage of resources that increase the risk of overload. As for DI values, we obtained higher risk of degradations which can be partly explained with a higher load applied to the system. The rest of reasons have to be found in the more realistic evaluation we could perform thanks to the use of realistic network and traffic data instead of a synthetic data plane used in simulations. Also in the laboratory tests we noticed that especially in the ND-FF case more links have a load within the $[80\%, 100\%]$ interval which increases the risk of data delivery degradations. However, in the experiments in laboratory we could not compute the number of overload but just qualitatively assessed through the pattern of link utilizations derived from the overall shape of the plots.

The set-up time measurements were taken during the experiments performed on a laboratory testbed. Though the obtained values had the same order of magnitude, the virtualized environment allowed us to have more accurate results (a much lower confidence interval). This is mainly because it offers a set of identical servers with the same capabilities which homogenizes the performance and makes the system more stable. In fact, in the laboratory tests we noticed that the duration of such operations depends on the server that is involved. This is due to the different hardware of the

workstations used in the experiment [8]. Moreover, as in the virtualized environment, in the experimental testbed, the computation time and the path set-up time have a negligible impact on the total time. On the other hand, the DHCP time and the cloning time are responsible for most of the total time necessary to complete the allocation of a new VM. In fact, such operations involve the interaction with the DHCP server and the cloud platform that are time-consuming actions.

The impact of the estimation parameters on the orchestration operation were also assessed during the simulations and the set of experiments performed on a laboratory testbed. The observed results in the virtualized environment confirmed the results obtained during the simulations, although with a more stable trend in simulations as expected thanks to the high number of iterations that can be performed under the same conditions and the large scale of the DC topology. Moreover, thanks to fully software-based operations to collect results in simulations we could use larger parameter ranges. A deviation in the trend was observed for DI measurements for different ΔT values. This could be explained by the use of synthetic data traffic in simulations. Regarding the experimental tests, because of the limited set of resources and the topology constraints, we could not draw conclusions about the overall performance of the effect of the estimations on the orchestration process. Instead, we selected a set of use cases under specific conditions (current load, monitoring parameters settings) and we measured the load of a subset of links. From this, we demonstrated the effect of the orchestrator allocation decisions and the estimation process on the resources management objectives (e.g., link usage consolidation). As result, we could carry out qualitative analysis in terms of advisability of an estimation process that smooths traffic spikes while tracking average load values versus a more reactive orchestration system that quickly reacts to traffic spikes although temporary that the final settings need to be decided based on the desired pattern of resource utilization, e.g., consolidation of resource usages, load balancing, along with the congestions risk the DC operator is willing to take.

V. Conclusions

This paper discussed pros and cons of using virtual testbeds to carry out Cloud-related experimentations while coping with significant scale requirements of DC infrastructures. More specifically, in this work we carried out experiments on the SDN and cloud resource orchestration while using a realistic DC environment to increase performance analysis soundness. This has been possible thanks to the use of the FED4FIRE virtualized testbed infrastructure. The SDN-DC orchestrator performance has been extensively evaluated in terms of an extended set of performance indicators and under realistic network set-up and traffic traces. The presented results confirm the feasibility and effectiveness of our proposed orchestration algorithms that outperform the baseline both in terms of request acceptance ratio, network efficiency and resources utilization. Indeed, thanks to the continuous monitoring and online estimation of the traffic load, the proposed SDN orchestration process ensures higher rate of VM allocations while guaranteeing a higher resource utilization of both servers and network capacity at the cost of a minimal risk of data delivery degradations. Moreover, the obtained results provided significant and additional inputs on how to tune the orchestration settings in order to obtain more efficient performance.

The experimental analysis also provided a comparison between virtual testbeds and other testing environments, i.e., laboratory testbeds, simulators and emulators, in terms of offered advantages and effectiveness in experimentations. Indeed, by using virtual testbed we could overcome some inherent restrictions of laboratory testbeds and simulation environments in terms of scale, reproducibility of actual system capabilities (e.g., real data packet switching across network nodes in simulators), accuracy of results and, finally, easiness of use and deployment. Indeed, virtual testbed environments are flexible, deployable and configurable in a straightforward way and allow to address also scale requirements of Cloud- and DC-related exerimentations. In addition, with respect to simulators, virtual testbeds can be set-up using the same software modules the experimenter is evaluating for a possible use in the field while leveraging a scale which is higher than the one used in the laboratory testbed. Overall, the experiment results confirm the trend observed in the simulations and laboratory testbed but with more accurate and extended evaluations thanks to both the realistic scale and operations we could achieve using the virtual testbed. In the near future, additional experimental activities are planned to evaluate the orchestration system operation while applying other selection algorithms (e.g., Best Fit) and in comparison with alternative approaches (e.g. round robin selection).

References

[1] C. N. Hofer *et al.*, "Cloud computing services: taxonomy and comparison," *Journal of Internet Services and Applications, Springer*, vol. 2, no. 2, pp. 69–79, September 2011.

[2] Y. Zhang *et al.*, "Evaluating the impact of data center network architectures on application performance in virtualized environments," in *Quality of Service (IWQoS), 18th International Workshop on*, June 2010.

[3] A. Lara *et al.*, "Network innovation using openflow: a survey," *IEEE Commun. Surv. Tutorials*, 2014.

[4] R. Guerzoni *et al.*, "Analysis of endtoend multidomain management and orchestration frameworks for software defined infrastructures: an architectural survey," *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 4, April 2017.

[5] R. Cziva *et al.*, "Sdn-based virtual machine management for cloud data centers," *IEEE Trans Netw Service Manag.*, 2016.

[6] "Ofelia official project site," http://www.fp7-ofelia.eu/.

[7] M. Gharbaoui *et al.*, "Cloud and network orchestration in sdn data centers: Design principles and performance evaluation," *Computer Networks*, 2016.

[8] D. Adami *et al.*, "An sdn orchestrator for cloud data center: System design and experimental evaluation," *Trans Emerging Tel Tech.*, 2017.

[9] https://www.ict-fire.eu/.

[10] https://www.fed4fire.eu/.

[11] B. Martini *et al.*, "Design and evaluation of sdn-based orchestration system for cloud data centers," in *Communications (ICC), IEEE International Conference on*, 2016.

[12] C. Guo *et al.*, "Dcell: A scalable and fault tolerant network structure for dcs," in *Proc. of the ACM SIGCOMM*, 2008.

[13] M. Alizadeh *et al.*, "Dc tcp (dctcp)," in *Proc. of ACM SIGCOMM*, 2010.

[14] S. Mehta *et al.*, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *IEEE Network Operations and Management Symposium (NOMS)*, 2008.

[15] M. Mihailescu *et al.*, "Optimized application placement for network congestion and failure resiliency in clouds," in *Proc. of IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015.

[16] W. Cerroni *et al.*, "Cross-layer resource orchestration for cloud service delivery: A seamless sdn approach," in *Computer Networks*, vol. 87, 2015, pp. 16–32.

[17] S. Fichera *et al.*, "On experimenting 5g: Testbed set-up for sdn orchestration across network cloud and iot domains," in *Network Softwarization (NetSoft), IEEE Conference on*, 2017.

[18] J. Chase *et al.*, "Joint virtual machine and bandwidth allocation in software defined network (sdn) and cloud computing environments," in *in Proc. IEEE ICC*, 2014.

[19] J. Son and R. Buyya, "Sdcon: Integrated control platform for software-defined clouds," in *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[20] A. Csoma *et al.*, "Escape: Extensible service chain prototyping environment using mininet, click, netconf and pox," in *ACM SIGCOMM*, 2014.

[21] B. Sonkoly *et al.*, "Unifying cloud and carrier network resources: An architectural view," in *In Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2015.

[22] A. Sgambelluri *et al.*, "Orchestration of network services across multiple operators: The 5g exchange prototype," in *2017 European Conference on Networks and Communications (EuCNC)*, 2017.

[23] http://www.geni.net/.

[24] J. Kang *et al.*, "Savi testbed: Control and management of converged virtual ict resources," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2013.

[25] J.-M. Kang *et al.*, "Software-defined infrastructure and the savi testbed," in *International Conference on Testbeds and Research Infrastructures. Springer, Cham*, 2014.

[26] B. Pataki *et al.*, "Sensor data collection experiments with chaoster in the fed4fire federated testbeds," in *IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2014.

[27] J. Becedas *et al.*, "The geo-cloud experiment: Global earth observation system computed in cloud," in *Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2014.

[28] M. Blaszczyska *et al.*, "medvc - a remote collaboration solution enhanced with cloud services and future internet capacities," in *eChallenges e-2015 Conference*.

[29] A. Abdelhadi *et al.*, "Position estimation of robotic mobile nodes in wireless testbed using geni," in *Annual IEEE Systems Conference (SysCon), Orlando, FL*, 2016.

[30] M. Gerola *et al.*, "Demonstrating inter-testbed network virtualization in ofelia sdn experimental facility," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013.

[31] H. Niavis *et al.*, "Auction-based scheduling of wireless testbed resources," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2014.

[32] https://aws.amazon.com. Accessed February 2019.

[33] https://cloud.google.com/.Accessed February 2019.

[34] R. A. Raton *et al.*, "Design and management of dot: A distributed openflow testbed," in *IEEE Network Operations and Management Symposium (NOMS)*, 2014.

[35] N. Handigol *et al.*, "Reproducible network experiments using container-based emulation," in *CoNEXT*, 2012.

[36] P. Wette *et al.*, "Maxinet: Distributed emulation of software-defined networks," in *IFIP Networking Conference*, 2014.

[37] http://jfed.iminds.be/.

[38] https://www.fed4fire.eu/jfed/.

[39] http://openvswitch.org/.

[40] P. H. Page, https://openflow.stanford.edu/display/ONL/POX+Wiki. Accessed March 2017.

[41] doc.ilabt.imec.be.

[42] https://www.fed4fire.eu/testbeds/virtual-wall/.

[43] https://wiki.xenproject.org.

[44] "iperf," https://iperf.fr/.

[45] van Adrichem NLM *et al.*, "Opennetmon: Network monitoring in openflow software-defined networks," in *Proc. of IEEE Network Operations andManagement Symposium (NOMS)*, 2014.

[46] M. Al-Fares *et al.*, "Hedera: dynamic flow scheduling for dc networks," in *Proc. of NSDI, CA*, 2010.

[47] C. Hopps *et al.*, "Analysis of an equal-cost multi-path algorithm," in *RFC 2992*.

[48] B. Martini *et al.*, "An sdn orchestrator for resources chaining in cloud data centers," in *Networks and Communications (EuCNC), 2014 European Conference on*, 2014.

[49] M. Gharbaoui *et al.*, "On virtualization-aware traffic engineering in openflow data centers networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014.

[50] https://www.isi.edu/nsnam/ns/.

[51] http://mininet.org/.

[52] https://www.labmate-online.com/news/news-and-views/5/breaking-news/what-is-the-difference-between-repeatability-and-reproducibility/30638. Accessed February 2019.

Barbara Martini is a Head of Research at the CNIT, Italy and Adjunct Professor at the Scuola Superiore SantAnna and University of Pisa. Her research interests include network virtualization and orchestration in SDN/NFV/5G environments, service platforms for next-generation networks, security for multi-domain networks, control/management architectures.

Molka Gharbaoui is a researcher at CNIT National Laboratory of Photonic Networks, Italy. She received her Ph.D. degree in Innovative Technologies of Information & Communications Engineering and Robotics in 2012 from the Scuola Superiore SantAnna, Pisa. Her main research interests are the support of QoS in transport networks, the development and the implementation of service oriented architectures, cloud computing and service management for smart cities.

Davide Adami is currently a senior researcher at CNIT, Italy. Previously, he worked at Consorzio Pisa Ricerche. He was involved in several Italian and EU research projects. His research interests mainly concern QoS in IP/MPLS networks, the design and development of new innovative solutions for the integration of Grid applications and networks architectures providing QoS support.

Piero Castoldi is a Full Professor and leader of the Networks and Services research area at the TeCIP Institute of Scuola Superiore SantAnna, Pisa, Italy. His research interests cover telecommunications networks and system both wired and wireless, and more recently reliability, switching paradigms and control of optical networks, including application-network cooperation mechanisms, in particular for cloud networking.

Stefano Giordano is a Full Professor at the Department of Engineering of the University of Pisa. He is in charge of the Telecommunication Networks Laboratories at the Department of Engineering of the University of Pisa. He was for many years in charge of the Networks at the Center Destination of Consorzio Pisa Ricerche, and is a member of the IEEE Communication Society since 1989.