Experimental Evaluation of Dynamic Resource Orchestration in Multi-layer (Packet over Flexi-Grid Optical) Networks

Silvia Fichera, Barbara Martini, Ricardo Martínez, Ramon Casellas, Ricard Vilalta, Raul Muñoz, Piero Castoldi

Experimental Evaluation of Dynamic Resource Orchestration in Multi-Layer (Packet over Flexi-Grid Optical) Networks*

S.Fichera[1], B. Martini[2], R. Martínez[3], R. Casellas[3], R. Vilalta[3], R. Muñoz[3], and P. Castoldi[1]

[1] Scuola Superiore Sant'Anna, Pisa, Italy
{s.fichera, castoldi}@santannapisa.it
[2] CNIT, Pisa, Italy
barbara.martini@cnit.it
[3] Centre Tecnòlogic de Telecomunicacions de Catalunya *(CTTC/CERCA)*, Barcelona, Spain
{rmartinez, rcasellas, rvilalta, rmunoz}@cttc.es

Abstract. In future 5G infrastructures, network services will be deployed through sets of Virtual Network Functions (VNFs) leveraging the advantages of both Software Defined Networking (SDN) and Network Function Virtualization (NFV). A network service is composed of an ordered sequence of VNFs, i.e., VNF Forwarding Graph (VNFFG), deployed across distributed data centers (DCs). Herein, we present a Cloud/Network Orchestrator which dynamically processes and accommodates VNFFG requests over a pool of DCs interconnected by a multi-layer (packet/flexi-grid optical) transport network infrastructure. We propose two different cloud and network resource allocation algorithms aiming at: i) minimizing the distance between the selected DCs, and ii) minimizing the load (i.e., consumed cloud resources) of the chosen DCs. Both algorithms run on a Cloud/Network Orchestrator and are experimentally validated and benchmarked on the CTTC ADRENALINE testbed.

Keywords: Resource Orchestration · SDN · NFV

1 Introduction

5G networks are being designed to leverage the inherited benefits brought by both softwarized networks and virtualization techniques (e.g., SDN and NFV). In this context, end-to-end 5G network services will be composed of an ordered set of interconnected network functions deployed in the cloud and running on commercial-off-the-shelf servers as virtual machines (i.e., VNFs). ETSI NFV defined an architectural framework for the coordinated VNF MANagement and

* This work is partially funded by the EU H2020 5G TRANSFORMER project (761536) and the Spanish AURORAS project (RTI2018-099178.

Orchestration (NFV MANO framework [1]) also across different hypervisors and computing resources deployed in remote cloud infrastructures (i.e., DCs) [2]. The resulting logical VNF topology accommodating a given end-to-end network service is defined as VNF Forwarding Graph (VNFFG) [3]. To this end, it is needed that the VNFs placed in remote DCs and their interconnections ensure QoS demands in terms of availability, bandwidth, latency also considering the current load of both cloud and network resources [4]. This entails in particular to properly select the network resources to provide DC interconnections, especially in the context of heterogeneous (packet and flexi-grid optical) transport infrastructures (i.e., Multi-Layer Network - MLN).

In this paper we present a cloud and network resource orchestration system and discuss the implications when dynamically serving VNFFG requests on top of the considered MLN. The orchestration of cloud and network resources is performed by an allocation engine (referred to as *Allocator*) complementing the ETSI NFV Orchestrator (NFVO) in the dynamic selection of the DCs and respective interconnections underpinning the VNFFG over a distributed DC infrastructure interconnected by a MLN [5]. Every DC is physically attached to packet switch nodes (e.g., MPLS), named as packet Gateways (Gws). Such Gws are equipped with sliceable bandwidth variable transceivers (SBVTs) providing the interconnection among them over a flexi-grid optical infrastructure. Upon receiving a new VNFFG request, the *Allocator* computes the cloud and networking resources and interacts with specialized controllers to yield the programmability and instantiation of such resources. In particular, for the networking resources the *Allocator* relies on a Transport SDN Controller (T-SDN) [6] enabling the resource computation and configuration of the underlying MLN infrastructure. In the adopted MLN scenario, it is worth outlining that opportunistic traffic *grooming* decisions are fostered to attain efficient use of network resources in terms of packet ports, SBVTs' transceivers and the optical spectrum.

In light of the above scenario, two on-line resource orchestration algorithms are proposed for selecting virtual resources: *(i)* compute resources at cloud DCs hosting VNFs and, *(ii)* network links enabling the VNF connectivity across the MLN. The key objective is to not only effectively deploy and satisfy VNFFG requirements but also accomplishing efficient use of the overall resources to favour serving subsequent VNFFG requests. This objective is generically integrated into the problem of embedding a virtual network onto a physical cloud and transport infrastructure which is referred to as Virtual Network Embedding (VNE) [7]. The VNE problem can be decomposed into two sub-problems: *virtual node mapping* and *virtual link mapping*. The ordering and criteria to execute them do impact on the attained resource utilization and, thereby on the resulting network service performance [8]. A number of VNE mechanisms have been proposed in the last years to optimally map virtual networks over a substrate infrastructure and guarantee end-to-end network services' requirements [18]. In a nutshell, these mechanisms address the optimal accommodation of virtual network demands from different perspectives, ranging from ensuring end-to-end QoS [9][10], economical targets [11], addressing survivability and energy-efficiency [12][14][13],

etc. Concerning flexi-grid optical networks, in [16], the problem of the interconnecting myriad of DCs by virtual networks on top of an elastic optical network has been tackled. Moreover, in [15], a combined orchestration process for both cloud and and network resources to interconnect multiple DCs over a packet and optical transport infrastructure has been proposed. As far as our knowledge, all these approaches focus on optimizing the resource consumption and do not consider the impact of the network delay which is becoming an important requirement when deploying network services across distributed DCs [17]. Moreover, the proposed solutions have been barely evaluated over experimental testbeds thus overlooking on the actual feasibility and efficiency of the algorithms in a real network deployment scenario.

In this paper the VNE problem is addressed by the two proposed resource orchestration algorithms which are experimentally evaluated within the CTTC ADRENALINE testbed [6] using a myriad of figures of merit. More specifically, the following metrics inspired by the VNE problems [18] are considered: the *path length* metric and the *stress level*. With the first metric, the targeted and stringent QoS requirement in terms of end-to-end latency can be effectively represented and quantified. In general, the larger is the connection path, the higher the experienced delay is. Thus, the first resource orchestration algorithm leads to select cloud DC resources which do minimize the resulting network delay when providing DC inter-connectivity. The (*stress level*) metric allows reflecting the DC occupancy. The second resource orchestration algorithm uses such a metric to prioritize a load balancing strategy for selecting the DCs hosting the required VNFs.

2 Resource orchestration set-up in a MLN Infrastructure

The considered cloud/network resource orchestrator system governing a pool of distributed DCs interconnected by a MLN is depicted in fig. 1. This constitutes the set-up (reproduced into the CTTC ADRENALINE testbed) used for realizing the experimental evaluation [5]. As mentioned the connectivity between a pair of DCs is done interconnecting respective packet Gws through a flexi-grid optical network. In this MLN scenario, *grooming* decisions are exploited to leverage the best of both packet and optical switching technologies: *i)* statistical multiplexing (i.e., mutiple packet traffic flows being transported over the same optical connection), and *ii)* huge transport capacity provided by optical transmission. The MLN is controlled by a T-SDN based on a PCE Central Controller (PCECC) [6]. Consequently, for an incoming network service requiring a packet connection between a pair of DCs hosting two VNFs, the T-SDN Controller triggers a constrained shortest path computation (CSPF) algorithm taking into account technological constraints and required service and performance objectives. To this end, the algorithm uses as inputs: *i)* the updated state of the network resources (i.e., packet ports utilization, optical spectrum availability, SBVT usage, etc.), *ii)* the network service requirements such as demanded bandwidth and maximum tolerated latency. It is worth outlining that eventual flex-grid

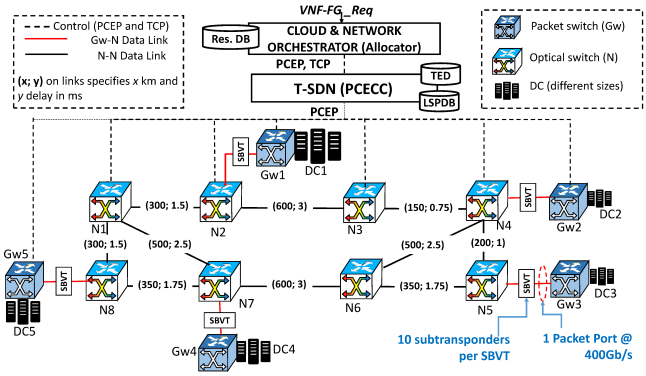


Fig. 1: Resource Orchestration on top of a MLN set-up

optical connections deployed between a pair of packet nodes (Gws) are derived on the so-called virtual packet link (VL). These VLs inherit attributes (e.g., available bandwidth, accumulated delay, etc.) from their underlying (optical) connections [6], and its spare available bandwidth can be considered by subsequent CSPF computations to accommodate new packet connections demands exploiting the targeted grooming opportunities.

The functions (i.e., network resource computation and MLN programmability) made by the T-SDN controller are coordinated by the cloud/network orchestrator (i.e., the *Allocator*). Specifically, the *Allocator* is the frontend for processing incoming VNFFG requests, checking DCs' resource availability, deciding for the proper selection of both cloud and network resources based on their load and availability, and finally allocating selected resources (assisted by the T-SDN controller as for network links). In other words, the *Allocator* is able to perform not only cloud/DC selection but also inter-DC path computations based on retrieved abstracted network information passed by the T-SDN controller. For the latter, the *Allocator* and the T-SDN Controller communicate among themselves using two APIs: i) standard Path Computation Element Protocol (PCEP) [22] for requesting path computation and/or instantiating feasible paths derived from the abstracted topology; ii) a proprietary TCP API for retrieving the abstract network information. To do the above, the *Allocator* relies on two repositories: *Cloud database* storing the DC cloud resource status and the *Traffic Engineering Database* (TED) storing the abstracted network topology (i.e., VLs between connected DCs)[4]. Recall that VLs are dynamically set up through the establishment of low-data rate packet connections over coarse flexigrid optical bandwidth connections. Based on the repositories' information, the

[4] . The MANO functions and the NFVO are not explicitly shown in the set-up, since their functions that are relevant in this work, i.e., VNFFG request processing and WIM function, are realized by the *Allocator*, that is particularly focused on the dynamic network resource selection and allocation process. Similarly, the DC infrastructure and the VIM functions have not been actually deployed yet emulated.

Allocator runs a particular on-line orchestration algorithm to properly select the (cloud and network) resources to address the VNFFG requirements while meeting specified goals, e.g., minimize the latency and balancing the DC resources load. A detailed description of the two proposed orchestration algorithms is provided in the following section.

3 On-line Resource Orchestration Algorithms

Without lack of generality, we assume that a VNFFG request is composed of interconnected VNFs deployed at two different DCs. The required VNFs are deployed over a set of virtual machines (VMs) supporting the demanded capabilities and capacities specified in the network service request. Therefore, for every VNFFG request (*req*), cloud resources are allocated into two DCs (i.e., *srcDC* and *dstDC*). Every *req* (fig. 2) specifies the number of VMs along with their computing requirements (i.e., CPU, RAM and Storage) per DC (*srcDC* and *dstDC*) as well as the inter-DC networking requirements: bandwidth (in Bytes/s) and maximum tolerated latency (in ms). The details of the proposed resource orchestration algorithms are discussed in the following:

VNFFG dynamic request generation	
Description	*srcDC* and number of VMs to allocate with IT (CPU.RAM and Storage) and Network (Bw and Latency) demand
Num *reqs*	1000; **Poisson** (mean interarrival time: 25s); **Exponential** duration (Holding Time. HT) varied from 200.250.300 and 350s
Num VMs per DC in a *req*	Uniformely distributed [1.5]
Resources per VM	Uniformely distributed: CPU [1.4]cores. RAM [1.6]GB. Storage [4.10.20.40]GB
Network Resources per Packet_LSP$_i$	**Bandwidth** unoformely distributes: 10. 40 and 100 Gb/s; **Latency**: uniformely distributed [6.12]ms

Fig. 2: VNFFG req generation details

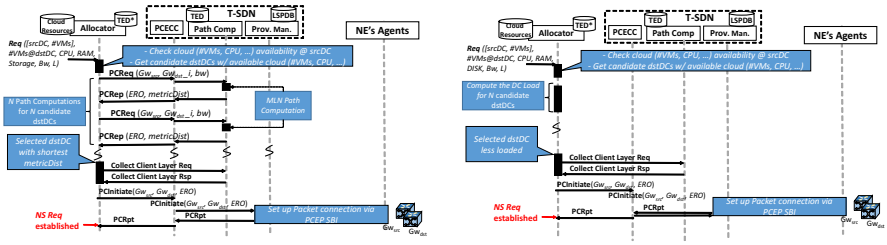
- *Minimum Distance* (MD): given the specified *srcDc* in the *req*, it selects the *dstDC* out of a candidate set of DCs resulting the closest in terms of distance (km) provided that: i) it has sufficient available cloud resources to serve the *req* and, ii) the inter-DC bandwidth requirement is fulfilled. By doing so, MD allows minimizing the experienced network delay due to the connectivity between both *srcDC* and *dstDC* leading to better deal with the *req*'s latency upper limit.
- *Less Loaded DC* (LLDC): given the *srcDC* in the *req*, LLDC chooses the *dstDC* out of a candidate set of DCs having the larger amount of available cloud resources (i.e., less loaded DC) provided that: i) it has sufficient available cloud resources to serve the *req*; ii) the inter-DC bandwidth requirement is fulfilled, and iii) the maximum required tolerated latency is not exceeded.

Fig. 3 shows the control workflows (i.e., interactions between the *Allocator* and the T-SDN Controller) required for MD and LLDC algorithms when serving a VNFFG request. Regardless of the algorithm, upon receiving a *req*, the *Allocator* verifies whether the *srcDc* has enough available computing resources. Additionally, it seeks for a subset of candidate *dstDCs* able to address the cloud resource demands using either MD or LLDC. If either conditions fails *req* is refused.

Focusing in the MD approach, in fig. 3(a), the *Allocator* sends to the T-SDN Controller a PCEP Path Computation Request (PCReq) message to compute a path from the *srcDC*'s Gw to all the possible *dstDCs*' Gws. The PCReq message carries the the Gws endpoints and the requested bandwidth (Bw). For each PCReq message, the T-SDN Controller triggers a K-CSPF algorithm (described in [5]) to satisfy both Bw and latency constraints. The K-CSPF algorithm aims at finding a feasible MLN path attaining the most efficient use of the network resources (i.e., packet ports, optical spectrum, S-BVT devices, etc.). If a path is found, the T-SDN Controller sends a PCEP Path Computation Reply (PCRep) message to the *Allocator* with the path (i.e., nodes, links, frequency slot, modulation format, SBVTs subtransponders) along with a metric value reflecting the actual distance (in km) between the Gw's endpoints. This metric then allows the *Allocator* selecting the *dstDC* with the lowest distance (in km). Afterwards, the *Allocator* addresses the allocation of the selected *dstDC* cloud and network resources over the pre-computed inter-DC path.

In the LLDC approach, in fig. 3(b), for each candidate *dstDC*, the *Allocator* computes the percentage of available cloud resources as a ratio between the amount of unused cloud resources over the sum of the whole deployed resources. Next, the *Allocator* selects the DC having the highest ratio, i.e., less loaded DC. Once the *dstDC* is chosen, the *Allocator* retrieves via the T-SDN controller the abstracted network view. Using the set of gathered VLs, the *Allocator* runs a shortest path route computation to connect both *srcDC* and *dstDC* Gws. The goal is to reuse as much as possible the spare available bandwidth of the existing VLs exploiting the benefits of grooming strategies. If this succeeds, the *Allocator* sends a PCEP PCInitiate message to the T-SDN controller with the computed route (i.e., set of VLs carried into Explicit Route Object, ERO) to perform the network programmability. Otherwise, the *Allocator* cannot find a feasible route (e.g., the set of VLs does not provide connectivity between the *srcDC* and *dstDC* Gws), and delegates to T-SDN controller the path computation over the MLN to provided the targeted DC connectivity.

In both resource orchestation strategies, if the DC connectivity succeeds, a PCEP Path Computation Report (PCRpt) message is sent back to the *Allocator*. Conversely, a Path Computation Error (PCErr) is sent to the *Allocator* informing that no connectivity between the selected *srcDC* and *dstDC* Gws is found, and thus the network service request is blocked.



(a) MD approach workflow (b) LLDC approach workflow

Fig. 3: Cloud/Network Orchestrator - T-SDN controller Workflows

4 Experimental Performance Evaluation

The performance evaluation of the proposed resource orchestration algorithms is conducted within the CTTC ADRENALINE testbed rolling out the cloud/network orchestrator and MLN infrastructure shown in fig. 1.

The cloud infrastructure is made up of 5 (emulated) DCs of different size (i.e., supporting different amount of cloud resources) which are connected to their corresponding 5 Gw nodes (i.e., MPLS switches). In particular, we consider 2 small DCs (40 CPU Cores, 160GB of RAM and 7TB of Storage) connected to both Gw2 and Gw3, 2 medium DCs (80 CPU cores, 320GB of RAM and 10TB of Storage) connected to Gw4 and Gw5 and 1 large DC (500 CPU cores, 2400GB of RAM and 135TB of Storage) connected to Gw1. Each Gw node has a single packet port (operating at 400Gb/s) connected to the optical flexi-grid network via an SBVT with 10 subtransponders. Each subtransponder can use 3 different modulation formats (MFs), i.e., DP-16QAM, DP-8QAM and DP-QPSK, enabling 3 different bit rates, i.e., 200, 150 and 100 Gb/s, for maximum distances, i.e., 650, 1000 and 3000 km, respectively. Optical links support 128 Nominal Central Frequencies spaced 6.25GHz. Optical fiber distances in fig. 1 are necessary for determining the MF when executing the K-CSPF computation as well as the accumulated path delay for checking the *req* latency restriction. Specific details of the *req* generation including the amount of demanded cloud resources and bandwidth and latency needs are described in fig. 2. In a nutshell, each experimental data point is realized with 1000 requests following a Poisson process VNFFG request whose mean inter-arrival time is set to 25 s, and the duration (holding time, HT) is exponentially modeled varying its mean to 200, 250, 300, and 350 s. This provides different offered traffic loads (expressed in Er): 8, 10, 12, and 14. The requirements of each VNFFG requests are generated as follows: the number of VMs per DC is uniformly distributed between [1,5]; the IT resources (i.e., CPU, RAM and disk) are randomly chosen in the ranges of [1,4] cores, [1,6] GB and [4,10,20,40] GB, respectively; the demanded bandwidth is randomly selected among [10,40,100] Gb/s; and the latency (l) is in the range of [6,12] ms.

Fig. 4 shows the percentage of the accepted *reqs* versus the offered traffic load in Erlangs (Er) when applying either MD or LLDC. We observe that higher

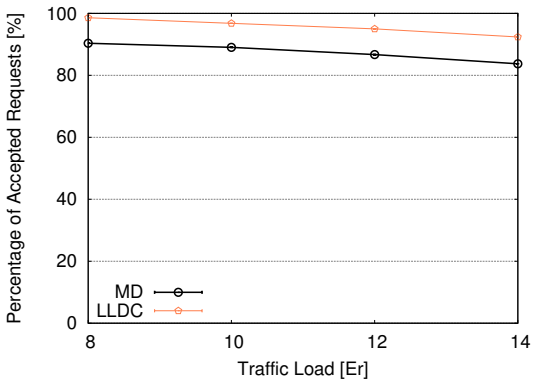
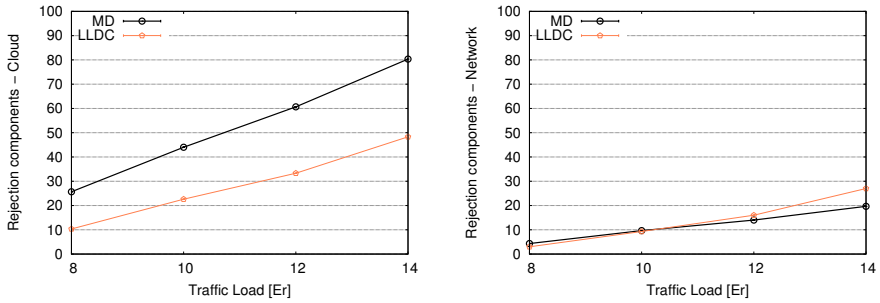


Fig. 4: Percentage of accepted requests vs. Traffic Load (Er)

rate of accepted *reqs* are attained by LLDC. This means that LLDC objective fostering to prioritize selected *dstDC* being less loaded leads to attain a more efficient use of all DC resources which in turn increases the chances to serve future network services. To consolidate this statement, it is thoroughly explored the reason why a *req* could be blocked. In fig. 5 we show the two main causes of requests rejection: lack of cloud resources at the DCs as well as the lack of network resources entailing the unavailability to satisfy the latency and/or bandwidth requirements imposed by each VNFFG request. In other words, for those VNFFG requests that cannot be accommodated, it is depicted the number of rejected requests per traffic load related to either of the above blocking causes. Fig. 5(a) shows that most of the blocked requests are due to the cloud resource unavailability in the MD case. This is because the selection of the *dstDC* is exclusively done by minimizing the distance with the *dstDC*. That is, no strategy providing efficient compute resource utilization among all DCs is applied leading to exhaust resources at specific DCs. On the other hand, fig. 5(b) shows that at lowest traffic load, the rejection due to the network unavailability is similar for both MD and LLDC. At a traffic load of 12 Er, the LLDC experiences more blocked requests caused by the network unavailability. Here the reason is the opposite with respect to the previous case wherein as traffic load grows the LLDC finds more problems to satisfy network requirements especially the latency requirement.

In fig. 6 the propagation delay experienced by data from *srcDC* and *dstDC* using the two resource orchestration approaches is depicted. As expected, MD lowers the obtained delay since it performs cloud and network resource computation aiming at minimizing the inter-DC path distance between both *srcDC* and *dstDC*. Consequently, MD always performs accomplishing the lowest propagation delay. Conversely, LLDC approach attains an accumulated propagation delay almost doubling the one achieved by the MD algorithm. As said, LLDC targets a better optimization of the DC resources whilst the distance between DCs (i.e., resulting propagation delay) is not minimized. Observe that as the



(a) Rejection due to Cloud unavailability (b) Rejection due to the Network availability

Fig. 5: Request rejection analysis

traffic load is increased, LLDC propagation delay is smoothly decreased. Indeed, as network service requests are increased, resources are more used and making more difficult to dealt with the network constraints. Aligned to the above, it becomes more complicated to satisfy the latency requirement for the incoming requests as shown in fig. 5(b). Therefore, in LLDC, at high traffic load, successfully established services tend to be deployed in shortest inter-DC paths, which entail shortest propagation delay.

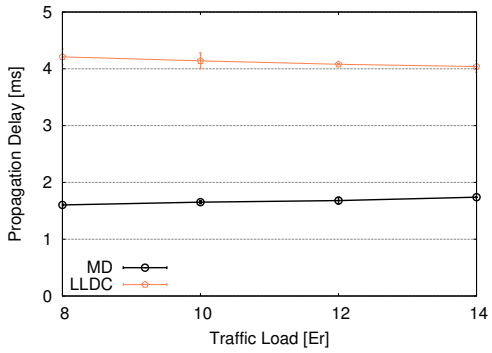


Fig. 6: Propagation Delay vs. Traffic Load (Er)

Fig. 7 plots the obtained Bandwidth Blocking Ratio (BBR). BBR provides the ratio between the amount of bandwidth being blocked and the total bandwidth being requested for all the received *reqs*. This figure of merit becomes relevant to show how well the K-CSPF algorithm performs when computing paths over the MLN infrastructure targeting grooming opportunities. In other words, exploiting grooming decisions leads to accomplish better use of the network resources which in turn does make lowering the BBR. That said, from the

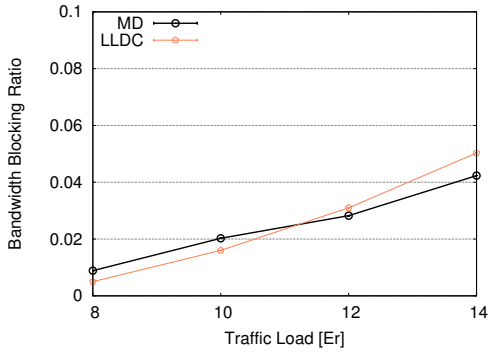


Fig. 7: Bandwidth Blocking Ratio vs. Traffic Load (Er).

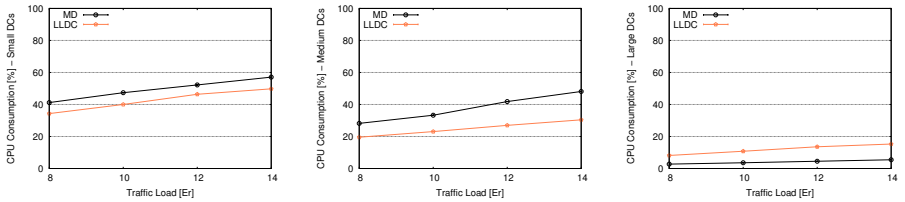
results one may realize that the resource orchestration algorithm applied in the *Allocator* when selecting *dstDc* has notable impact on the obtained BBR. In this regard, observe that as traffic load is increased MD performs better (i.e., lower BBR) when compared to LLDC. The reason behind that is as traffic load grows, network resources tend to be more occupied. This complicates the K-CSPF to find feasible MLN paths (when needed) satisfying the set of imposed technological constraints such as spectrum continuity and contiguity. Consequently, adopting a resource orchestration strategy which minimizes the distance between *srcDC* and *dstDC* leads to use, in general, less network resources. Consequently, MD selection facilitates the K-CSPF algorithm to deal with those mentioned technological constraints.

In fig. 8 the amount of CPU being allocated for each DC size is shown for the two approaches. MD tends to allocate more VNFs in the small and medium DCs. In fig. 8(a) the MD approach occupies almost the 60% of the CPU available in the small DCs when HT = 14 Er whereas the LLDC approach reaches the 50%. This behaviour is also seen for medium DC (fig. 8(b)). However, this trend is reversed for large DC (fig. 8(c)) where the MD and LLDC occupy 10% and 20% of the CPU, respectively. In fact LLDC prioritizes allocating cloud resources in DC having more available resources, which uses to be the larger DCs.

Finally, in fig. 9 it is depicted the average set-up time, i.e., the overall time required to: i) select the *dstDC*, ii) compute the inter-DC connectivity, and iii) allocate the network resources of such an inter-DC connectivity (either reusing VLs or allocating new MLN resources). As expected from the above discussed workflows, MD approach requires longer time to set up the network services mainly due to the amount of control interactions between the *Allocator* and the T-SDN Controller to derive the shortest distance for each candidate *dstDC*.

5 Conclusions

In this paper, we compared two on-line cloud and network resource orchestration algorithms (MD and LLDC) to dynamically accommodate network services



(a) CPU consumption in small DCs (b) CPU consumption in medium DCs (c) CPU consumption in large DCs

Fig. 8: CPU consumption in different DC sizes

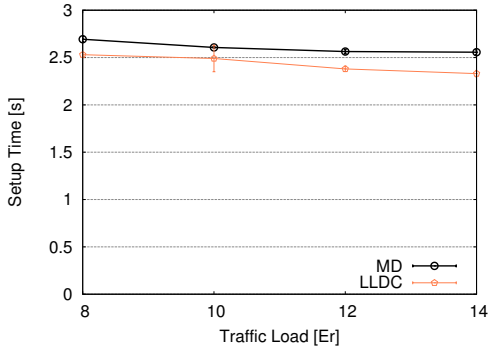


Fig. 9: Setup time vs. Traffic Load (Er)

(expressed as VNFFGs) within distributed remote DCs being inter-connected through a MLN infrastructure. The performance evaluation of both algorithms has been done experimentally within the CTTC ADRENALINE testbed using a myriad of figures of merit such as the acceptance ratio, the BBR, the consumed CPU resource per DC size, etc. using different traffic loads associated to the amount of generated VNFFG requests. The MD algorithm aims at minimizing the inter-DC connectivity distance between the selected DCs to lower the resulting end-to-end latency. On the other hand, LLDC is devised to attain a more efficient use of the compute resources throughout all the DCs. In light of the obtained results, one can state that LLDC algorithm does improve the network service request acceptance thanks to the beneficial effect of balancing the compute resource load at the expenses of increasing the end-to-end latency.

References

1. ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration", Group Specification, 2014
2. ETSI, "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services", ETSI GR NFV-IFA 022 V3.1.1, 2018

3. ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification", ETSI GS NFV-IFA 005 V2.1.1, 2016
4. B. Martini and F. Paganelli, "A service-oriented approach for dynamic chaining of virtual network functions over multi-provider software-defined networks", Future Internet, vol. 8, no. 2, pp. 24, 2016.
5. S. Fichera, *et al.*, "Experimental Evaluation of Orchestrating Inter-DC Quality enabled VNFFG Services in Packet/Flexi-Grid Optical Networks", in proc. *of ECOC*, Sept. 2018.
6. R. Martínez, *et al.*, "Experimental Evaluation of a PCE Transport SDN Controller for Dynamic Grooming in Packet over Flexi-Grid Optical Networks", in proc. *of ECOC*, Sept 2017.
7. A. Haider, *et al.*, "Challenges in Resource Allocation in Network Virtualization", in *20th ITC Specialist Seminar*, vol. 18,2009.
8. M. Gharbaoui, *et al.*, "Cloud and network orchestration in SDN data centers: Design principles and performance evaluation", *Computer Networks*, vol. 108, 2016.
9. T. Trinh, *et al.*, "Quality of service using careful overbooking for optimal virtual network resource allocation," The 8th Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) Association of Thailand - Conference 2011, pp. 296-299.
10. X. Zhang, *et al.*, "An overlay mapping model for achieving enhanced QoS and resilience performance," 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Budapest, 2011.
11. M.R. Rahman and I. Aib, "Survivable Virtual Network Embedding", in *Networking*, 2010, pp. 4052.
12. J. Shamsi and M. Brockmeyer, "Efficient and dependable overlay networks," IEEE International Symposium on Parallel and Distributed Processing, 2008.
13. J. F. Botero,*et al.*, "Energy Efficient Virtual Network Embedding," in IEEE Communications Letters, vol. 16, no. 5, pp. 756-759, May 2012.
14. G. Sun, *et al.*, "The Framework and Algorithms for the Survivable Mapping of Virtual Network onto a Substrate Network," 2011 IETE Technical Review.
15. B. Kong, *et al.*, "Demonstration of application-driven network slicing and orchestration in optical/packet domains: on-demand vDC expansion for Hadoop MapReduce optimization," Opt. Express 26, 2018.
16. Z. Zhu, *et al.*, "Build to tenants' requirements: On-demand application-driven vSD-EON slicing," in IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 2, Feb. 2018.
17. B. Martini, *et al.*, "Latency-aware composition of virtual functions in 5G", Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft), Apr. 2015
18. A. Fischer, *et al.*, "Virtual Network Embedding: A Survey", *IEEE Communications Surveys Tutorials*, vol.15, 2013
19. https://cloudify.co/
20. https://openbaton.github.io/
21. https://osm.etsi.org/
22. JP. Vasseur *et al.*, "Path Computation Element (PCE) Communication Protocol (PCEP)", in IETF RFC 5440, March 2018.