Multi-Cloud Chaining with Segment Routing

Francesco Spinelli[♯], Luigi Iannone[†], Jerome Tollet[⋆]
[♯]IMDEA Networks Institute & Universidad Carlos III de Madrid, Madrid, Spain,
[†]LTCI, Telecom Paris, Institut Polytechnique de Paris, France,
[⋆]Cisco Systems Inc, Issy-les-Moulineaux, France,
E-mails: [♯]francesco.spinelli@imdea.org, [†]luigi.iannone@telecom-paristech.fr, [⋆]jtollet@cisco.com

***Abstract*—In recent years, next to Cloud Computing, Network Function Virtualization (NFV) has emerged as one of the most interesting paradigms inside the ICT world, leading to unexplored scenarios and new features such as Service Chaining. The latter consisting in steering the packets through a sequence of services on their way to the destination. Even more, these services can be located inside different Public Clouds, hence giving us the possibility to exploit for the first time a *Multi-Cloud* approach. One way to perform Service Chaining is through the new Segment Routing protocol for IPv6. Within this context we have investigated how we could create, inside Amazon Web Services, a Multi-Cloud configuration to provide Service Chaining, leveraging on the Vector Packet Processing (VPP) software router for packet handling and SRv6 processing. We performed extensive measurement campaigns, looking in particular on how VPP and Segment Routing presence could affect the overall performance inside Amazon Web Services and having a first insight on what performance can be achieved when chaining several cloud instances in different geographical regions.**

***Index Terms*—Segment Routing, Service Chaining, Virtual Private Clouds, AWS, NFV, VPP**

I. Introduction

Nowadays, the implementation of Software Defined Networks (SDN) and Network Function Virtualization (NFV) in real infrastructures, together with Cloud Computing, allows to explore both new scenarios and new paradigms such as *Multi-Cloud* usage and Service Chaining [1], [2]. We refer to *Multi-Cloud* as the use of multiple cloud computing and storage services in a single heterogeneous architecture. This is also known as a Polynimbus [3] cloud strategy. This also refers to the distribution of cloud assets, software, applications, etc. across several cloud-hosting environments. With a typical multi-cloud architecture utilizing two or more public clouds as well as multiple private clouds, a multi-cloud environment aims to eliminate the reliance on any single cloud provider. There are a number of reasons for deploying a multi-cloud architecture, including reducing reliance on any single vendor, cost-efficiencies, increasing flexibility through choice, adherence to local policies that require certain data to be physically present within the area/country, geographical distribution of processing requests from physically closer cloud unit which in turn reduces latency, and mitigating against disasters. A customer may want to chain services from different cloud providers because they are different or complementary. Or else, a customer may want to avoid cloud provider lock-in and prefers to buy the same service from different providers.

ISBN 978-3-903176-28-7© 2020 IFIP

Creating a Private Cloud chaining services offered in different public clouds is a complex task involving several aspects, namely: placing, instantiating, and connecting. As a first building block, there is a need to actually have the means to control the network traffic, so to drive it through the right sequence of services. This is actually the problem we tackle in this paper. To this end, we leverage on Segment Routing for IPv6 [4]. This new protocol, from now on SRv6, allows a source node to steer a packet through an ordered list of segments, encoded as IPv6 addresses. This list is stored inside the new Segment Routing Header (SRH), which is part of the IPv6 header [5]. Every segment is associated with a function placed at a specific location in the network. A function could be for instance a VNF sitting in a different public cloud. We decided to use this protocol especially for its similarity with the Service Chaining's principle.

We were the first ones in exploiting SRv6 inside a Public Cloud Provider infrastructure in a preliminary version of the work presented in this paper [6]. Here we provide a more extensive evaluation inside the Amazon Web Services Cloud infrastructure [7], comparing the IPv6 traffic vs IPv4 and adding a more complex scenarios, including connecting three different cloud regions in a chain. The results obtained from our measurements campaigns, show that the performance penalties, on inserting the Segment Routing per IPv6's Header inside the packets, compared to a native IPV4 traffic flow, are for the majority of the cases irrelevant, thus making the new Segment Routing protocol a perfect candidate to bring Service Chaining inside public clouds. We also evaluated throughput and latency as a function of the geographical distance among different AWS cloud instances.

One of our paper's novelty is to bring Service Chaining in a real Cloud infrastructure. Most of the previous works in the related field instead focus on a more theoretical approach ([8][9][10][11]), with some exceptions such as in [12], where the authors build an experimental platform, [13], [14], and [15]. Even though the idea of using Segment Routing for Service Chaining is not entirely new ([16][17] [18][19]), we are the first ones in deploying and measuring the SRv6 performance using VPP inside a *Multi-Cloud* configuration).

The remaining of the paper is organized as follows. In Sec. II, we describe how we setup SRv6 service chaining in the Amazon Cloud. Then in Sec. III we introduce the methodology

used to evaluate our proposal, while in Sec. IV we provide the results we obtained. Sec. V concludes the paper.

II. AWS Environment Set-Up

Taking a look among all the Public Cloud Providers, we have decided to perform our deployment and experiments inside Amazon Web Services (AWS) Cloud environment [7] for two main reasons. The first reason is because AWS implements a feature called *Virtual Private Clouds (VPC)* [20], which allows the user to define a virtual network logically isolated from the others, with benefits on security, easiness of deployment, flexibility, and scalability [20]. Besides, when creating a VPC, the user could specify a whole set of characteristics such as, for example, the IP addresses range to allocate, subnets, firewalls and routing tables. The second reason is that AWS offers the possibility to allocate more than one IPv6 address per Network Interface Card (NIC), giving us the opportunity to use the SRv6 implementation of Vector Packet Processing (VPP [21]), which is at the moment constrained on using at least two IPv6 address per NIC.

VPP has been chosen because one of the goals we are also interested in is achieving the best possible performance. Indeed, VPP is a framework for building high-speed data plane functionalities in software, taking advantages of general-purpose CPU architectures. It is written in C and it is completely defined in user space. It exploits kernel-bypass techniques, leveraging for instance on DPDK [22], and its main feature is the effective processing of batch of packets using techniques such as *Multi-Loop*, *Data prefetching* and *Direct Cache Access*, among the others [21]. In our configuration and experiments we used VPP version 18.07. AWS gives the possibility to allocate a range of different hardware, depending on the user's goal. In our case, we decided to use the general purposes *m5.2x large type* for all of our VMs, hence, having nominally available maximum network bandwidth of 10 Gbps.

To automate the deployment of VPP and all of the resources necessary to successfully build our configuration inside AWS, we have decided to use Terraform, a Cloud Orchestrator tool ([23], [24]). It exploits the paradigm of *Infrastructure as Code*, allowing to describe the components of a cloud provider, such as VPCs, subnets, instances and route tables through a proprietary programming language called *HashiCorp Configuration Language*. Moreover, Terraform is *Cloud Agnostic*, which means that the same script, slightly modified, can be reused also with other Cloud Providers, enabling the possibility to bring our configuration in different Cloud environments.

Fig. 1 shows, at high-level, what exactly has been deployed in each AWS Virtual Private Cloud (VPC):

VPP instance: The VPP instance was used as SRv6 endpoint, hence, providing SRv6 encapsulation/decapsulation operations, as well as simple forwarding when needed.

Client/Server: The Client and Server virtual machines just represented the services to be chained.

Fig. 1 shows the high level view of the topology we created in our experiments, connecting together three different Public Cloud Regions. In this case the packets, before reaching their final destination, are steered towards another in-the-middle Public Cloud Region. We used two different setups: in the first one, all regions are inside one continent (namely North America), in the second one, every region is in a different Continent (North America, Europe, Asia). We also used a simplified scenario where composed of only two public cloud instances, lacking the instance in the middle, creating a point to point topology, in which we connected a maximum of two Public Cloud Regions belonging to the same Cloud Provider (AWS). The AWS Regions explored are Paris, London, Ireland, Oregon, Sao Paolo, Tokyo, Sidney.

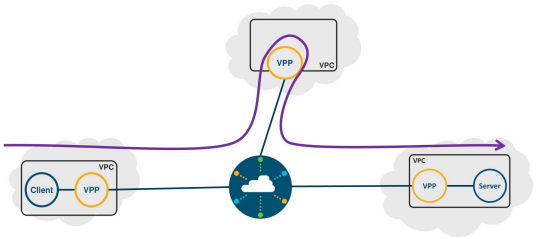

Fig. 1. High level view of topology used in our evaluation.

III. Evaluation Methodology

We performed several measurement campaigns, deploying our configuration inside several Amazon's Regions, comparing the impact of using VPP and SRv6 with respect to the native AWS network stack.

We focused on two metrics: namely throughput for both TCP Reno and UDP, measured using *Iperf3* [25], and the Round-Trip Time (RTT), measured using *Ping*. Unless differently stated, measurements consist of 30 runs lasting *20* seconds, performed on the client side and their average comes with a 95% of confidence interval, therefore gathering relevant statistical accuracy. In our configuration, the IP traffic flows from the client to the first VPP machine. Here the packets are processed by the software router and encapsulated with a Segment Routing Header. Afterwards, packets go as normal IPv6 packets through the Internet, towards the next public Cloud instance, if it is the final destination, packets are decapsulated and sent to the Server machine, otherwise the SRv6 header is updated so to forward the packet on the next segment. As already stated, as a baseline to measure the impact of using VPP and SRv6, we take also into account the case without the VPP, therefore sending the traffic directly among public cloud instances.

The regions' choice is driven by both the distance and age of construction of the corresponding data-center. We deployed VPP and our configuration inside seven different Amazon's region (Paris, London, Ireland, Oregon, Sao Paolo, Tokyo, Sidney) obtaining nine different combinations, since we also perform several Intra-Cloud measurements where the two VPCs are inside the same Region.

IV. Evaluation Results

We present here our experimental performance measurements. We first describe general results in the point-to-point

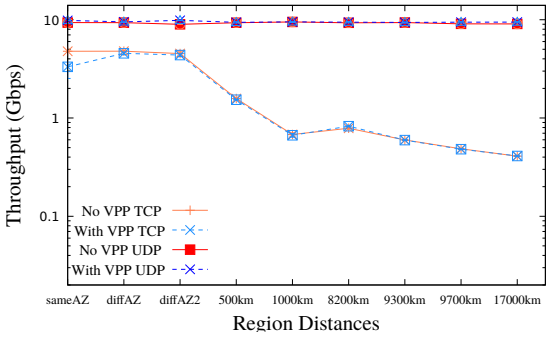

Fig. 2. Throughput evaluated with measurements lasting 20 seconds.

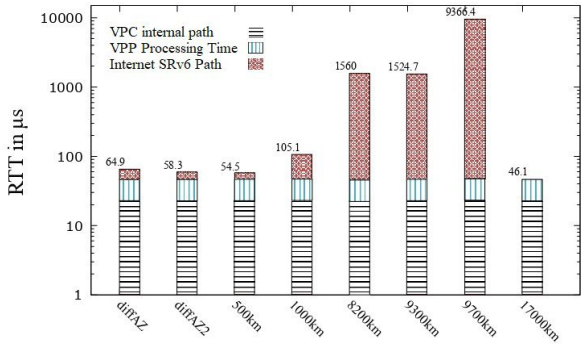

Fig. 3. VPP overhead in μs.

topology using IPv4 traffic (cf. Sec. IV-A).[1] Afterwards, we compare IPv4 vs IPv6 traffic in the same simple scenario (cf. Sec. IV-B), before providing the results for the more complex scenario with three chained cloud regions (cf. Sec. IV-C).

A. Point-to-Point Topology

In Fig. 2 we portrait the throughput, for both UDP and TCP traffic. In the x-axis we have put the distance km when the two VPCs are in different geographical regions. Inside an Amazon region, there could be different Availability Zones, meaning that exist several data-centers, each one located in a different and isolated part of the region. This gives the possibility to deploy resources in multiple data-centers, therefore exploiting reliability. In our figures, *SameAZ* means that the two VPCs are inside the same data-center inside the same Region (Paris), while with *DiffAZ* the VPCs are in two different data-centers but located in the same Region (both data-center being in London), as for the *DiffAZ2* case, where both the data-center are located inside the Tokyo Region. As the figure shows, in the UDP scenario, both with and without VPP, we were able to saturate the link at 10 Gbps. This is not the case for the TCP scenario. Amazon uses a shaper that slows down the TCP flow rate at maximum 5 Gbps, a behavior also noticed by Lai et al. [26]. Moreover, the throughput has two different behaviors, depending on the distance between source and destination. Firstly, the more the two VPCs become distant from each other, the more the throughput for both types of flows decreases, however, no significant difference can be noticed. This is explained by the TCP throughput dependency on RTT. Secondly, when the two VPCs are closer, for instance in the same region, we notice that the throughput evaluated without VPP clearly outperforms the one using VPP. This is justified by the overhead, even if very small, introduced by the presence of the VPP module.

We further explored the VPP overhead. Fig. 3 shows the difference between RTTs caused by the presence of the VPP module. Actually, only a small fraction of this difference consists in real VPP processing time (the part with vertical lines), the bulk (horizontal lines) is caused by the different path packets' follow when flowing through VPP. In fact, in a very deep level, Amazon imposes a specific path inside the VPC. When packets are sent from the Client Virtual Machine, they have to go first to the VPC's Internet Gateway, which will steer the packets towards the VPP machine. Here, they are encapsulated with the SRv6 and finally they are forwarded outside of the VPC, through the VPC's Internet Gateway. Mirror-like, when they arrive at the destination VPC, they first arrive at the VPC Internet Gateway, who forwards them to the VPP instance. The latter decapsulates the packets and sent them back to the VPC Internet Gateway, who finally forwards them to the Server. Instead, if we don't put any VPP nodes in the middle of the path, the packets sent from the Virtual Machine goes directly to the Internet Gateway and steered outside of the VPC. Fig. 4 shows the different path taken by the packets depending on the VPP presence.

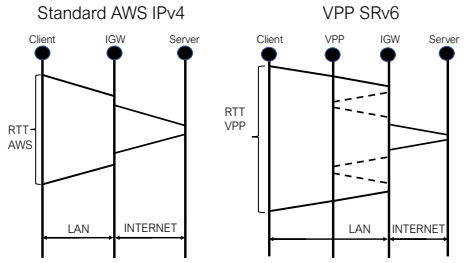

Fig. 4. Packets physical path inside an Amazon Virtual Private Cloud. The dashed lines represent the longer path due to the use of VPP.

In few cases, when the two VPC are particularly distant from each other, it also happens that SRv6 traffic through Internet follows completely different and longer paths when compared to IPv4 traffic (*8200km*, *9300km*, *9700km* cases inside Fig. 3). Nevertheless, while we are adding two more hops in the path (the VPP instances) and a latency penalty due to SRv6 routing in the Internet [19], performance remains very very close to what is possible to obtain with native IPv4 traffic, but with the additional benefit of being able to chain services in different VPCs.

[1]Note that SRv6 is defined only for IPv6, hence, when IPv4 traffic is generated by the client, it is encapsulated in IPv6 with the SRv6 extension header. However, when IPv6 traffic is generated by the Client, VPP only needs to add the SRv6-specific extension header.

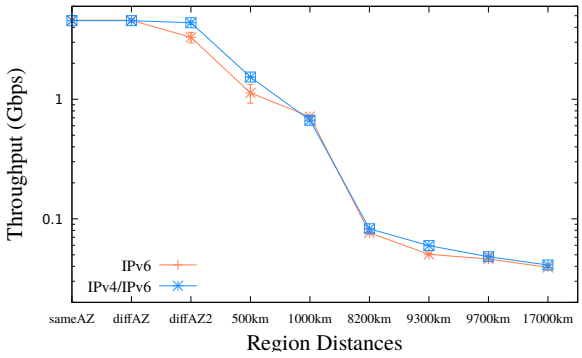

Fig. 5. Throughput evaluated with measurements lasting 20 seconds with a 95% confidence interval. The regions are the same explored in Sec. IV-A.

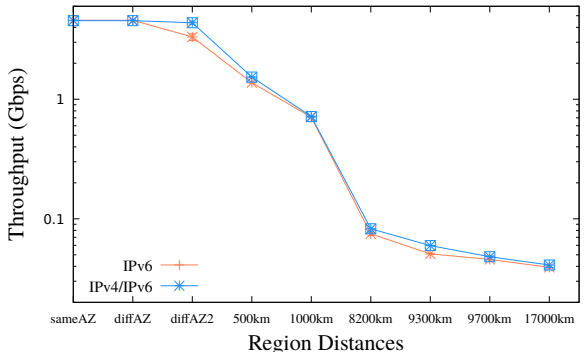

Fig. 6. Throughput obtained inverting the flow direction among the same AWS instances.

B. IPv4/IPv6 vs all-IPv6 traffic

Because of the use of SRv6, we wanted to dig a little bit deeper about the usage of IPv6. In fact, nowadays IPv6 traffic is steadily increasing together with the cost of maintaining IPv4 addresses, therefore ISPs are migrating more and more from IPv4 to IPv6 [27]. AWS gives the opportunity to create and use an IPv6 only infrastructure. For the above reasons, we began another measurements campaign, looking at the difference between an all IPv6 scenario and the old IPv4/IPv6 scenario, which is shown in Fig. 5. For coherence, we explored the same scenario of Fig. 2 but we decided to show only the TCP throughput, since it seemed to us more relevant. From Fig. 5 we can see that in both cases the throughput has a similar behavior. Only in few cases the all-IPv6 throughput is slightly smaller, noticeable for *DiffAZ2*, when both VPCs are inside the Tokyo region, for *500km* (when the server is in London), and for *9300km* (when the server is in Sao Paulo).

Using exactly the same settings we performed another measurement, but in this case we inverted the flow direction. We were interested in seeing if the throughput was symmetric or not, since in literature it has been proven that single TCP flows in AWS are not necessarily symmetric [26]. Fig. 6 shows the obtained result, where we cannot observe a huge difference between the two cases, except for *500km*, with the Client machine in London and the Server machine in Paris, where the all-IPv6 throughput reduces its gap comparing to the IPv4/IPv6 throughput.

We went deeper, trying to find why a gap is present between the two measurements (IPv4/IPv6 vs all-IPv6). Hence, we measured the RTT for both flows, looking in particular if we could find a difference between the RTT values, such as we show in Fig. 3. In Fig. 7 we show our results. It is possible to notice that the values are similar, expect for some parts, namely *DiffAZ2, 500km, 9300km*, leading us to confirm that in these cases the RTT difference is the reason of the gap between the evaluated throughput.

C. Chained Topology

For our last scenario, we elaborate two more complex topologies, in order to forward the packets through a real

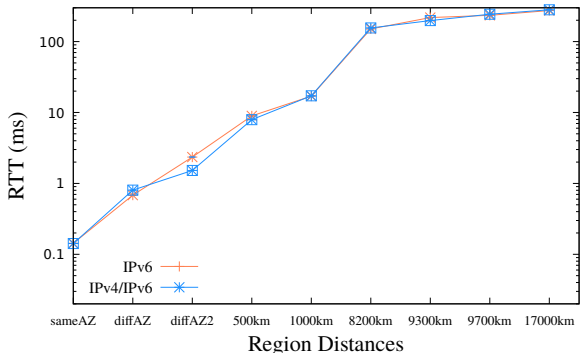

Fig. 7. RTT of IPv4-over-IPv6 traffic vs all-IPv6 traffic.

Service Chaining architecture. We explored two scenarios: *i*) all regions belonging to a single continent (namely North America);*ii*) every region is inside a different continent. In our opinion this was the best way to see how SRv6 would affect the network infrastructure and performance. like in the previous sections, we performed our measurements using both IPv4/IPv6 traffic and IPv6-only traffic.

Fig. 8 shows the results and we can see two important points: first, again in both scenarios the throughput achieved for both traffic type is very similar and second that when the VPCs are in regions belonging in different continents, the throughput dramatically decreases. However, this is a behavior we expected since with a longer path the RTT is bigger.

In Fig. 9 we portrait the throughput with the inverted flow, exchanging the sending machines, as we did in Fig. 6. In this case we noticed that the traffic is not bi-directional, as already highlighted in [26]. On the contrary, the all-IPv6 traffic throughput in the same continent is dramatically decreased, even more than a half (we were not able to identify the reason of such drastic decrease in throughput). On the contrary when the used regions belonging in different continents, the throughput has slightly augmented.

V. Conclusion

In this paper we presented the initial results of our research in bringing Service Chaining in a Multi-Cloud environment.

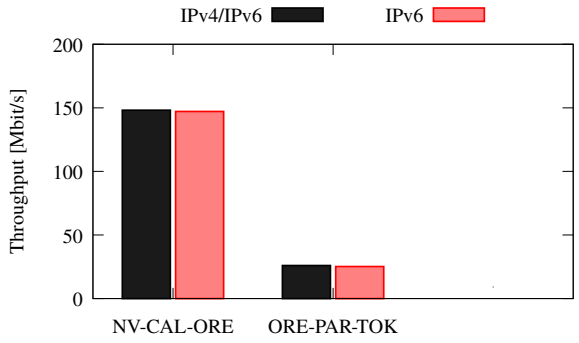

Fig. 8. Throughput with three different regions. The regions exploited are: New Vegas, CALifornia, OREgon, PARis, TOKyo.

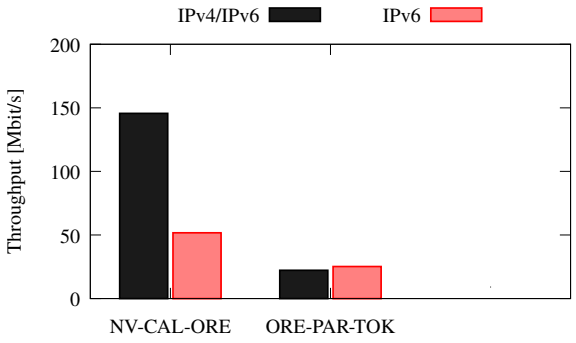

Fig. 9. Throughput with the same three regions as in Fig. 8, but with inverted flows.

We have deployed and tested SRv6 inside a Public Cloud Provider, being the first ones, to the best of our knowledge, in performing Service Chaining inside AWS. Our measurements, show that, even with a slight penalty, SRv6 protocol performs well inside AWS, making it a good candidate for Service Chaining inside the Public Cloud Providers domain.

As future work we will explore possibility is bring our solution inside other Public Cloud Provider such as *Microsoft Azure* and *Google Cloud*, trying also to connect different Cloud Provider together. This would lead to even more complex *Multi-Cloud* topologies, therefore providing the scenario to perform more in-depth measurements, taking into consideration also how different services could impact the performance.

Acknowledgment: The work presented in this article benefited from the support of NewNet@Paris, Cisco's Chair "Networks for the Future" at Telecom ParisTech (https://newnet.telecom-paristech.fr). Any opinions, findings or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of partners of the Chair.

REFERENCES

[1] R. Buyya and J. Son, "Software-defined multi-cloud computing: a vision, architectural elements, and future directions," in *International Conference on Computational Science and Its Applications*. Springer, 2018.

[2] D. Petcu, "A panorama of cloud services," *Scalable Computing: Practice and Experience*, vol. 13, no. 4, pp. 303–314, 2012.

[3] C. Pietschmann, "The Polynimbus Cloud Enterprise," 2016, https://buildazure.com/2016/12/23/the-polynimbus-cloud-enterprise.

[4] C. Filsfils, Z. Li, J. Leddy , D. Voyer, D. Bernier, F. Clad , P. Camarillo,. (2018, Mar) SRv6 Network Programming.

[5] Clarence Filsfils et al. (2018, Mar) Ipv6 segment routing header (srh). [Online]. Available: http://tools.ietf.org/html/draft-ietf-6man-segmentrouting-header

[6] F. Spinelli, L. Iannone, and J. Tollet, "Chaining your virtual private clouds with segment routing," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2019, pp. 1027–1028.

[7] Amazon Web Services. [Online]. Available: https://aws.amazon.com

[8] D. Bhamare, R. Jain, M. Samaka, G. Vaszkun, and A. Erbad, "Multi-cloud distribution of virtual functions and dynamic service deployment: Open adn perspective," in *2015 IEEE International Conference on Cloud Engineering*. Tempe, AZ, USA: IEEE, March 2015, pp. 299–304.

[9] "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Computer Communications*.

[10] P. Wang, J. Lan, X. Zhang, Y. Hu, and S. Chen, "Dynamic function composition for network service chain: Model and optimization," *Computer Networks*, vol. 92, pp. 408 – 418, 2015.

[11] T. Kuo, B. Liou, K. C. Lin, and M. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *The 35th IEEE International Conference on Computer Communications IEEE INFOCOM'16)*, San Francisco, CA, USA, April 2016.

[12] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo, "Towards making network function virtualization a cloud computing service," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Ottawa, ON, Canada: IEEE, May 2015, pp. 89–97.

[13] P. Quinn and J. Guichard, "Service function chaining: Creating a service plane via network service headers," *Computer*, vol. 47, no. 11, pp. 38–44, Nov 2014.

[14] M. B. Anwer, M. Motiwala, M. b. Tariq, and N. Feamster, "Switchblade: a platform for rapid deployment of network protocols on programmable hardware," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, Aug. 2010.

[15] A. Bremler-Barr, Y. Harchol, and D. Hay, "Openbox: Enabling innovation in middlebox applications," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMiddlebox '15. New York, NY, USA: ACM, 2015, pp. 67–72.

[16] F. Clad and et al, "Segment routing for service chaining," Internet Engineering Task Force (IETF), Tech. Rep., oct 2017, work in Progress.

[17] A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a linux-based nfv infrastructure," in *2017 IEEE Conference on Network Softwarization (NetSoft)*. Bologna, Italy: IEEE, July 2017, pp. 1–5.

[18] P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils, "Sdn architecture and southbound apis for ipv6 segment routing enabled wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1378–1392, Dec 2018.

[19] D. Lebrun and O. Bonaventure, "Implementing ipv6 segment routing in the linux kernel," in *Proceedings of the Applied Networking Research Workshop*, ser. ANRW '17, 2017.

[20] AWS. (2019) Amazon web series vpc documentation. [Online]. Available: https://docs.aws.amazon.com/en_us/vpc/latest/userguide/what-is-amazon-vpc.html

[21] D. Barach, L. Linguaglossa, D. Marion, P. Pfister, S. Pontarelli, and D. Rossi, "High-speed software data plane via vectorized packet processing," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 97–103, December 2018.

[22] DPDK. [Online]. Available: https://www.dpdk.org/

[23] HashiCorp. (2019) Introduction to terraform. [Online]. Available: https://www.terraform.io/intro/index.html

[24] V. Docs. (2018, may) Vpp in aws. [Online]. Available: https://fdio-vpp.readthedocs.io/en/latest/usecases/vppinaws.html

[25] Iperf. (2019) Iperf web site. [Online]. Available: https://iperf.fr/

[26] F. Lai, M. Chowdhury, and H. Madhyastha, "To relay or not to relay for inter-cloud transfers?" in *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*, Boston, MA, 2018.

[27] I. Society. (2018, June) State of ipv6 deployment 2018. [Online]. Available: https://www.internetsociety.org/resources/2018/state-of-ipv6-deployment-2018/