Chainage de vos Virtual Private Clouds avec Segment Routing

Francesco Spinelli[1] et Luigi Iannone[1] et Jerome Tollet[2]

[1]*LINCS, Telecom ParisTech, 23 avenue d'Italie, 75013 Paris*
[2]*Cisco Systems Inc, 11 Rue Camille Desmoulins, 92130 Issy-les-Moulineaux*

Au cours des dernières années, à coté du Cloud Computing, la virtualisation des fonctions de réseau (NFV) est devenue l'un des paradigmes les plus intèressants du monde des ICT, donnant lieu à des scénarios inexplorés et à de nouvelles fonctionnalités telles que Service Chaining. Ce dernier est la capacité où les paquets sont dirigés à travers une séquence de services sur leur chemin vers la destination. Ces services pourraient également etre situés dans différentes régions de cloud public, nous permettant ainsi d'exploiter pour la première fois une approche multi-cloud. Le nouveau protocole de routage par segment pour IPv6 (*Segment Routing – SRv6*) est un moyen efficace d'enchainer les services. Dans ce contexte, nous avons commencé à étudier comment créer une configuration multi-cloud dans Amazon Web Services afin de fournir un chainage de services. Dans un premier temps, nous avons déployé, au sein d Amazon Virtual Private Cloud, un routeur logiciel compatible avec le routage par segment appelé *Vector Packet Processing* (VPP). Ensuite, nous avons automatisé son déploiement avec Terraform, un outil pour d'orchestration. Ensuite, profitant de nos scripts, nous avons répété la meme configuration dans diffèrentes régions d'Amazon, en les connectant ensemble via le SRv6. Enfin, nous avons lancé une première campagne de mesures étendue, en examinant en particulier l'impact de la présence de VPP et SRv6 sur les performances globales d'Amazon Web Services.

Mots-clefs : Segment Routing, Cloud Computing, Virtual Private Clouds, AWS,IPv6, VPP, NFV, Software Routers.

1 Introduction

Nowadays, the implementation of Software Defined Networks (SDN) and Network Function Virtualization (NFV) in real infrastructures gives us the possibility to explore both new scenarios and new paradigms such as *Multi-Cloud* configurations and Service Chaining. The latter is becoming more and more popular because it is changing how to handle traffic flows, automating the setting up of virtual network connections [cen]. Instead, we refer to *Multi-Cloud* as the possibility to connect two, or more, Public Cloud Regions together, even belonging to different Cloud Providers, with benefits on both security and performance [LB17]. We started to explore these new possibilities, showing the results from our first measurements campaign using a simple Multi-Cloud configuration. In Figure 1 we show at high level the topology we used for our initial experiments: we have deployed a simple point to point topology, in which we connect a maximum of two regions belonging to the same Cloud Provider. Recently, with the development of Segment Routing per IPv6 [C. 18], a new way to perform Service Chaining has risen. This new protocol, from now on SRv6, allows a source node to steer a packet through a 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 [ea18]. Every segment is associated with a function called at a specific location in the network. A function could be for instance a VNF like a firewall or just an action like decapsulation or forwarding of the packet. Furthermore, SRv6 provides also the possibility to exploit *Quality of Service*, allowing to steer packets through paths with constraints in throughput or latency. Since we are also interested in performance, we decide to use Vector Packet Processing (VPP) to bring SRv6 inside AWS. VPP is a framework for building high-speed data plane functionalities in software [L. 18], 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

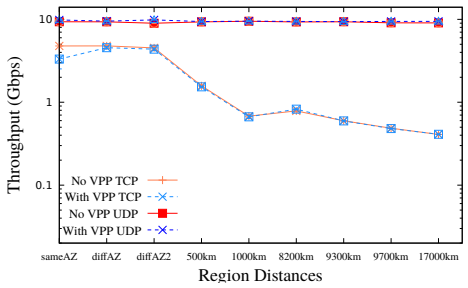


Figure 1: Simplified view of our initial topology. The AWS Regions explored are *Paris, London, Ireland, Oregon, Sao Paolo, Tokyo, Sidney.*

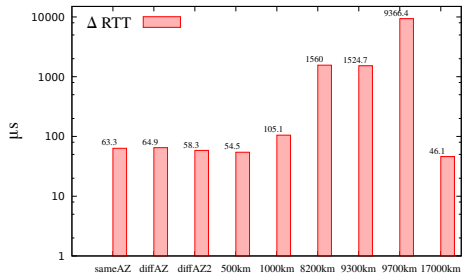
for instance on DPDK and Netmap, and its main feature is the effective processing of batch of packets using techniques such as *Multi-Loop*, *Data prefetching* and *Direct Cache Access*, among the others [L. 18]. In our setup we used VPP version 18.04. Taking a look among all the Public Cloud Providers, we have decided to perform our deployment and experiments inside Amazon Web Services (AWS) [AWSa] Cloud environment for two main reasons. The first reason is because Amazon implements a feature called *Virtual Private Clouds* [AWSb], which allows the user to define a virtual network logically isolated from the others with benefits on security, easiness of deployment and flexibility. In fact, 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 route tables. In Figure 1 we show, at high level, that in each VPC we deploy two virtual machines: *i)* a VPP machine, which allows the SR connectivity; *ii)* a Client or Server machine (representing the service to be chained). 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 VPP's SRv6 implementation, which is at the moment constrained on using at least two IPv6 address per NIC. Furthermore, to automate the deployment of VPP inside AWS, we have decided to use Terraform [Has], a cloud orchestrator tool. 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 could be reused also with other Cloud Providers, allowing us the possibility to bring our script in different Cloud environment. We made our Terraform script available to the community together with the instructions needed to create our environment [vpp18].

2 Related Work

Even though we are the first ones in measuring the SRv6 performance using VPP inside a *Multi-Cloud* configuration, in literature already exist some cases of SRv6 applications and measurements. For example, Abdelsalam et al. [ACF+17] work focuses on chaining VNFs which are SR-aware and SR-unaware, showing also a methodology for evaluating the performance. Another example is inside Ventre et al. [VTSF18] paper, in which they describe an SDN based approach for controlling SRv6 enabled networks. Finally, Lebrun et al. [LJC+18] implement in their paper a Software Resolved Network, which is an architecture similar to SDN made for enterprise networks. It is based on SRv6, since allows network operators to specify policies that control the network paths. Instead, focusing on the performance measurements, Lebrun et al. [LB17] work propose a first implementation of SRv6 inside the Linux kernel. Afterwards, they perform some measurements to evaluate their implementation, showing that SRv6 obtain 10% less performance with respect to plain IPv6 [LB17]. Furthermore, in [AVM+18], Abdelsalam et al. present a performance evaluation framework called SRPerf, which is able to evaluate different KPIs of Segment Routing such as throughput and latency. In their work, they noticed that in some cases there is a significant drop of SRv6 performance, compared to plan IPv6.



(a) Throughput evaluated with measurements lasting 20 seconds.For simplicity, with *sameAZ* and *diffAZ* we mean that the two VPCs are inside the same Region (hence we have maximum distance of $\approx$ 100km).



(b) VPP overhead in μs.

Figure 2: Throughput and VPP overhead evaluation.

3 Performance Experiments

For our measurements campaign, we deployed our configuration inside several Amazon's Regions, with the goal of evaluating what is the impact of using VPP and SRv6 compared to the native AWS network stack. Our experimental scenario consists in two different VPCs located either in the same region, or in two different regions, as highlighted in Figure 1. We are interested in three metrics: Throughput, with both TCP and UDP protocols, Round Trip Time (RTT) and Time To Live (TTL). These metrics are evaluated using the *Ping* and the *Iperf3* tools. The measurements take place on the client side and their mean comes with a 95% of confidence interval, therefore gathering relevant statistical accuracy. The traffic flows from the client to the first VPP machine, where the SRv6 protocol steers the packets towards the second VPP machine, and finally to the Server. Furthermore, we take also in account the case without VPP machine, sending the traffic directly from the Client to the Server, using IPv4 addresses. Our main hub is the Paris Region, which therefore is always our Client side. The choice of the regions is driven by both the distance and age of construction of the corresponding data-center. We deploy 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. Furthermore, AWS gives the possibility to allocate a range of different hardware, depending on the user's goal. We decided to use the general purposes *m5.2xlarge type* for all of our Virtual Machines, obtaining hence a maximum network speed of 10 Gbps. In Figure 2a we portrait the throughput evaluated with measurements lasting *20* seconds. In the UDP scenario, in both cases the two flows, characterized by the VPP presence or not, saturate the link at 10 Gbps, even though with VPP we achieve a slight gain overall. Instead, inside the TCP scenario we have obtained different results. Amazon uses a shaper that slows down our TCP flows rate at maximum 5 Gbps. Moreover, the throughput has two different behaviours. Firstly, the more the two VPCs become distant from each other, the more the throughput for both types of flows decreases. This is explained by the TCP throughput dependency on RTT. Secondly, when the two VPCs are closer we notice that the throughput evaluated without VPP clearly outperforms the one using VPP. This is justified by the overhead introduced by the presence of the VPP module. The latter, shown in Figure 2b, is caused in a small part by the processing time of VPP, but it is mainly caused by the different path taken by VPP packets inside the VPC. In fact, when a packet is ejected from the client Virtual Machine, it has to go firstly to the VPP machine, where it is encapsulated in an IPv6 packet and then finally it goes outside of the VPC. Afterwards, when it arrives at the destination VPC, it has to do again the previous steps, resulting in a longer RTT. In few cases, when the two VPC are particularly distant from each other, it also happens that SRv6 performs very different paths through Internet from the IPv4 path.

4 Conclusion

In this paper we presented only initial results of our research. Firstly, we have developed a script, available to the community, which helped us in automatically build our topology. Secondly, we have deployed and tested Segment Routing per IPv6 inside a Public Cloud Provider, being the first ones, to the best of our knowledge, performing Service Chaining inside AWS. As future work, we will investigate more complex Multi-Cloud configurations and also make more in-depth measurements. We will also explore the possibility to bring our solution inside other Public Cloud Provider such as *Microsoft Azure* and *Google Cloud*.

Acknowledgments: The work presented in this article has been carried out during Francesco Spinelli's internship at LINCS (https://www.lincs.fr/). It 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

[ACF+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)*, pages 1–5, July 2017.

[AVM+18] A. Abdelsalam, P. L. Ventre, A. Mayer, S. Salsano, P. Camarillo, F. Clad, and C. Filsfils. Performance of ipv6 segment routing in linux kernel. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 414–419, Nov 2018.

[AWSa] AWS. Amazon web series. available at *https://aws.amazon.com*.

[AWSb] AWS. Amazon web series vpc documentation. available at $https://docs.aws.amazon.com/en_us/vpc/latest/userguide/what-is-amazon-vpc.html$.

[C. 18] C. Filsfils, Z. Li, J. Leddy , D. Voyer, D. Bernier, F. Clad , P. Camarillo,. SRv6 Network Programming. *draft-filsfils-spring-srv6-network-programming-04.txt*, Mar 2018.

[cen] SDX central. What is network service chaining? definition. available at *https://www.sdxcentral.com/sdn/network-virtualization/definitions/what-is-network-service-chaining/*.

[ea18] C. Filsfils et al. Ipv6 segment routing header (srh), Mar 2018. available at *http://tools.ietf.org/html/draft-ietf-6man-segmentrouting-header*.

[Has] HashiCorp. Introduction to terraform. available at *https://www.terraform.io/intro/index.html*.

[L. 18] L. Linguaglossa, D. Rossi, S. Pontarelli, D Barach, D. Marjon, P. Pfister. High-speed Software Data Plane via Vectorized Packet Processing extended version. *Technical report*, May 2018.

[LB17] D. Lebrun and O. Bonaventure. Implementing ipv6 segment routing in the linux kernel. In *Proceedings of the Applied Networking Research Workshop*, ANRW '17, New York, NY, USA, 2017. ACM.

[LJC+18] David Lebrun, Mathieu Jadin, FranÃ§ois Clad, Clarence Filsfils, and Olivier Bonaventure. Software resolved networks: Rethinking enterprise networks with ipv6 segment routing. In *SOSR'18: Symposium on SDN Research*, 2018.

[vpp18] How to deploy vpp in ec2 instance and use it to connect two different vpcs, may 2018. available at $https://wiki.fd.io/view/How_to_deploy_VPP_in_EC2_instance_and_use_it_to_connect_two_different_VPCs$.

[VTSF18] 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*, 15(4):1378–1392, Dec 2018.