Analyzing the Performance of SD-WAN Enabled Service Function Chains Across the Globe with AWS

Aris Leivadeas
aris.leivadeas@etsmtl.ca
École de Technologie Supérieure
Montreal, QC, Canada

Nikolai Pitaev
npitaev@cisco.com
Cisco Systems, Inc.
San Jose, CA, USA

Matthias Falkner
mfalkner@cisco.com
Cisco Systems, Inc.
Ottawa, ON, Canada

ABSTRACT

Cloud Computing has revolutionized the information technology world and the application offering over the last two decades. At the same time recent trends in Network Function Virtualization (NFV) and Software-Defined Wide Area Networks (SD-WAN) and the combination of those with the Cloud paradigm has allowed an unprecedented shift of enterprise networking services towards the Public Cloud. Even though this network evolutionary approach brings many benefits, it still presents many drawbacks as well. The performance stability and service continuity over a black box Public Cloud infrastructure can hinder the formal service guarantees that many new emerging applications may have. To this end, in this paper, we aim to shed light on the overall performance achieved when deploying coast-to-coast and intercontinental Service Function Chains (SFCs) that interconnect geographically distributed enterprise branches over the Amazon Web Services (AWS) infrastructure. In particular, we investigate the impact of region, Virtual Machine (VM) instance, time of the day and day of the week in the overall throughput and delay attained. The obtained results show the strengths and weaknesses of entirely relying on the AWS infrastructure to offer networking services by investigating possible hidden performance bottlenecks.

CCS CONCEPTS

• **Networks** → *Cloud computing*; *Network servers*; *Network experimentation*; *Network performance analysis*; *Network measurement*.

KEYWORDS

NFV, VNF Performance, SD-WAN, AWS

ACM Reference Format:
Aris Leivadeas, Nikolai Pitaev, and Matthias Falkner. 2023. Analyzing the Performance of SD-WAN Enabled Service Function Chains Across the Globe with AWS. In *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3578244.3583722

ICPE '23, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0068-2/23/04…$15.00
https://doi.org/10.1145/3578244.3583722

1 INTRODUCTION

Cloud Computing offers a unique opportunity for a large range of actors to benefit from low cost service deployment to multi-tenancy and remote collaboration. Thus, lately, service providers (SPs), application providers, enterprises and common end-users resort to Public Cloud services. In fact, the use of Public Cloud steadily grows over the last decade, especially for enterprises that largely rely on popular providers (i.e., AWS, Azure, Google Cloud, etc.) to offer applications to their personnel and clientele [32].

At the same time, the advent of Network Function Virtualization (NFV), allowed the disassociation of networking functions from dedicated hardware appliances, that can be now offered as softwarized images called Virtual Network Functions (VNFs) [21]. The VNFs, can be deployed in a location-free manner, as Virtual Machines (VMs) or containers in generic servers. Networking services can be also composed of many interconnected and ordered VNFs that form Service Function Chains (SFCs) [22] that will offer the final service (i.e. security, video streaming, etc.) to be consumed.

NFV and Cloud Computing are highly correlated, since the core of both technologies is the virtualization mechanism. Thus, the networking department of many enterprises opt to offer their services as cloud-based services separated from their local IT departments and data centers (DC). The benefits of such a deployment are multifold. For example, many enterprise branches can share the same cloud-managed network service [24], leading to operational and capital expenses reductions. Additionally, different enterprise customers will be serviced by the same cloud-enabled networking service allowing multi-tenancy and further cost reductions.

Another technology considered in this paper is Software-Defined Wide Area Networks (SD-WAN). SD-WAN deploys software-defined techniques to cover sites that span across a wide geographical area [34]. A typical SD-WAN use case is the interconnection of different networks, DCs, or branches of an enterprise that are geographically dispersed. A centralized SD-WAN controller can manage the interconnection of these sites in a flexibly and performance-oriented manner. SD-WAN can benefit from NFV and Cloud towards providing fully programmable networking services scattered around the Globe according to the location of the different enterprise sites.

Thus, a new opportunity arises for enterprise, SPs, and Cloud Service Providers (CSP) to reap the benefits of an inter-disciplinary network paradigm. The mix of advantages introduced by this combination is immense and well-established for each individual technology. However, these advantages can be counterbalanced by the overall network performance attained, which in a Public Cloud infrastructure can highly depend on space and time. This motivated us to create a realistic cloud-enabled enterprise communication environment, by deploying SFCs in AWS, which is currently the

most popular Public CSP. Furthermore, the SFCs are managed by a SD-WAN controller that guarantees the smooth and agile interconnection of different enterprise sites scattered across the world. Our goal is to measure the performance achieved in terms of bandwidth and delay, when enterprise branches are interconnected through a SFC and exchange data in various time-frames during the day. Additionally, we aim to analyse the benefits and trade-offs in terms of performance and deployment cost and pinpoint any limitations that such a deployment may have. To the best of our knowledge, this is the first time that such an approach has been implemented and tested. The contributions of our work are as follows:

(1) We implement and test for the very fist time a SFC that is managed by a SD-WAN controller over AWS. The SD-WAN configures the SFC to offer security, data privacy and routing services, while enabling the collaboration and information exchange between enterprise branches and networks.
(2) We deploy the SFC in an inter-DC fashion examining two cases. In the first one, we create a coast-to-coast interconnection between US West and US East. In the second one, we increase the coverage to an intercontinental level between US West and Sydney, Australia.
(3) We rely on an AWS proprietary network solution, by making use of the new Amazon Transit Gateway (TGW) [4]. TGW Peering uses AWS backbone and avoids using the Public Internet, decreasing thus the security threats, increasing the performance of an intermittent and best-effort connectivity, and facilitating the deployment of SD-WAN solutions.
(4) We perform our experiments in different times of the day and days of the week. This approach allows us to investigate the impact of both space (different regions) and time on the overall performance obtained, by examining the variability of throughput and delay.
(5) We provide a discussion on the opportunities and disadvantages of such a solution emphasizing on the trade-off between cost and performance.

The remainder of the paper is structured as follows. Section 2, highlights the related work regarding performance analysis in Public Clouds. Section 3 provides the configuration of our deployment, explaining the main AWS functionalities and components used. Section 4 presents the performance analysis and discusses the obtained results. Finally, Section 5 concludes the paper and provides possible future directions.

2 RELATED WORK

Relying on virtualization to offer networking services has attracted a lot of attention the past few years. Even though network virtualization promises significant cost reductions, the total cost of ownership (TCO) calculations can be adversely impacted by the overall performance achieved in a virtualized system. In a Private Cloud or an enterprise DC, several software and hardware solutions can be followed to increase or guarantee the performance. For example, different resource footprint configurations [8], Input/Output (I/O) architectures [28], and packet accelerators [13] may be considered. However, in a Public Cloud, modifications and/or optimizations of the available systems and hardware are not possible. Additionally, CSPs only divulge qualitative information about network performance, creating thus operational uncertainties [26].

Hence, there is a substantial effort in the pertinent literature to quantify the performance in Public Clouds through different experimentation scenarios. These scenarios usually cover intra-DC communications [16, 29], inter-DC communications [14, 15, 27, 35], both intra and inter-DC communications [17, 23, 30], or cloud-to-user latency [26]. In this Section, we mostly emphasize on the performance achieved for intra and inter-DC connectivity.

For example, the authors in [29] studied the performance of VM instances in terms of CPU, I/O, and network in an intra-DC in AWS. Results showed that the performance can vary a lot, especially when compared with a similar deployment method in a local DC. A more NFV-related analysis was performed in [16], without however deploying SFCs. Specifically, the authors evaluated three different VNFs, namely Firewall, Intrusion Detection System, and NAT, when deployed in different AWS EC2 instances in the same region/DC. Once again, the performance showed great differentiation in terms of packet rate, packet loss, and resource utilisation according to the size of the EC2 instance used. Regarding inter-DC connectivity, the authors in [14] analyzed the delay achieved between all different combinations among 10 AWS regions. Results revealed that there was no significant variation in delay, regardless the time of the day and the time of the week. Concerning bandwidth, a number of throughput measurements between various DCs in both AWS and Azure platforms were performed in [15]. The analysis demonstrated that there was low variation on the measured throughput. A similar comparison between the AWS and Azure platforms was presented in [27]. The results showed that inter-DC throughput performance is better in Azure than Amazon, whereas the latency performance is comparable. An additional observation was that larger VM sizes do not always lead to higher performances. One reason for this behavior, is that the bottleneck lies on the interconnection paths between the DCs and not on the computational capacities of the VMs. This drove the authors in [35] to assess the performance of three different inter-DC connection types, namely the transit provider-based best-effort public Internet (BEP), the third-party provider-based private (TPP) connectivity, and the CSP-based private (CPP) connectivity over AWS, Azure, and Google Cloud Platform. As in our paper, they concentrated in a US coast-to-coast deployment and analyzed the performance in terms of throughput and delay. The results obtained showed the superiority of CPP in comparison to BEP and TPP and the reduction of the performance variability.

Finally, a number of existing works compare the performance of intra-DC against the inter-DC connectivity. For instance, the authors in [17] analyzed the performance of bandwidth-intensive applications in inter and intra-DC AWS settings. Particular emphasis was given on the Public Cloud policies that can impact the network performance and deployment cost. The results showed that larger VM sizes do not always increase the performance, while the inter-DC throughput achieved is always lower than the intra-DC one. These throughput findings were also corroborated by a more systematic analysis, where four different CSPs, namely AWS, Microsoft Azure, Google AppEngine, and Rackspace were considered for storage, computation and latency bound applications [23]. Furthermore, the performance of segment routing in terms of throughput and round trip time (RTT) was examined within and between different

Table 1: Comparison of related work

Reference	SFC	inter-DC	Private Connectivity	Different Time/Day	Different Instances	SD-WAN
[29]				✓	✓	
[16]					✓	
[14]		✓		✓		
[15]		✓		✓		
[27]		✓		✓	✓	
[35]		✓	✓	✓		
[17]		✓		✓	✓	
[23]		✓		✓	✓	
[30]	✓	✓				
Ours	✓	✓	✓	✓	✓	✓

AWS regions in [30]. The analysis of this deployment, which resembles to a simple routing service chain, revealed that the throughput can vary significantly with regards to the transport protocol used and the distance between the two regions interconnected.

All the above works are analyzing the performance by usually deploying only two VMs; one acting as a server and the other one as a client. Even though this approach is very informative for the performance achieved it is not a realistic Cloud deployment. In contrast, in our approach, we have a series of 6 VMs distributed in various regions that form a SFC of 4 VNFs, while the first and last VM act as the source and destination of the communication. Additionally, only the authors in [35] take into consideration a realistic enterprise scenario of a coast-to-coast inter-DC deployment, while leveraging a Cloud-based private connectivity. Similar to this work, we also consider such a deployment, extending it though to an intercontinental connection, while our SFCs are SD-WAN enabled. We also analyze the performance achieved in different instances of the week, while considering various sizes of VM instances. To better illustrate the contributions of our approach, Table 1 summarizes the key differences of our paper with the aforementioned studies.

3 DESIGN AND EXPERIMENTAL SETUP

In this Section we present the main deployment and measurement methodology followed in our work. Firstly, we begin by providing the overall architecture for the two inter-DC scenarios under consideration. Secondly, we introduce the main AWS components used to effectuate the provisioning of inter-DC SFCs. Thirdly, we show the software and tools used to deploy the different VNFs and SD-WAN components. Finally, we lay out the measurement methodology for generating the traffic and for gathering the results.

3.1 Overall Architecture

The main goal of this paper is to analyze the performance achieved when interconnecting different DC/regions on AWS to deploy SFCs. As such, we have analyzed two possible inter-DC connections. In the first one, a US coast-to-coast deployment is followed, as shown in Fig. 1. This is a typical multi-cloud deployment that a US enterprise may adopt [35]. In particular, we strategically position two virtual private clouds (VPCs) in two regions that their interconnection will traverse the USA. In the first VPC (in Oregon), we deploy a sender traffic generator instance and two VNFs. In the second VPC (in Ohio), we deploy two additional VNFs (mirrored VNFs with regards to Oregon's site) and a receiver traffic generator. These VNFs are selected to provide realistic use-cases for providing an SD-WAN enabled SFC (such as backup services, secure access service edge - SASE, etc.).

The interconnection of the VPCs is facilitated through AWS's own inter-region infrastructure. The SD-WAN can be found in the first region (i.e., Oregon) and manages and interconnects the two VPCs. In reality, the VPCs can be viewed as the two cloud-enabled enterprise branches. The control information of SD-WAN is sent through the Public Internet (dashed arrows) and not through the private's AWS intra/inter-region peering connection (blue arrows).

For the second scenario, we adopt a more particular use-case, where the two branches are found in different continents. In this case, we wanted to test how the performance will be affected in a transpacific context. In more detail, as can be seen in Fig. 2 we keep one region in Oregon, while the second is found in Sydney, Australia. The same SFC as the first scenario is considered, while the SD-WAN controller remains in Oregon. All the necessary information regarding the configuration of the SFCs, SD-WAN and various AWS components are provided in the following subsections.

3.2 AWS Components

In Figs. 1 and 2 we make use of the VPC to establish an intra-DC communication and AWS TGW for an inter-DC communication. In this part of the Section, we provide additional details for these two components and how they can be used for providing enterprise Cloud-based connectivity.

3.2.1 Virtual Private Cloud: Amazon VPC [1] offers a private and isolated virtual network in AWS. This can generate a unique opportunity for enterprises, since they can create their private DCs in a Public Cloud environment. Additionally, the enterprise can continue to have the control of the computational and communication resources of the VPC, while adding the necessary security and isolation. VPCs can be created in different regions and availability zones of AWS according to the location requirement of the enterprise. Regarding computational resources, the users can benefit from the available EC2 instances that can willingly create in their own VPC. With regards to communication resources, VPC allows the creation by default of up to 200 subnets with different CIDR ranges [2], conforming to the exigencies of our use case. Finally,

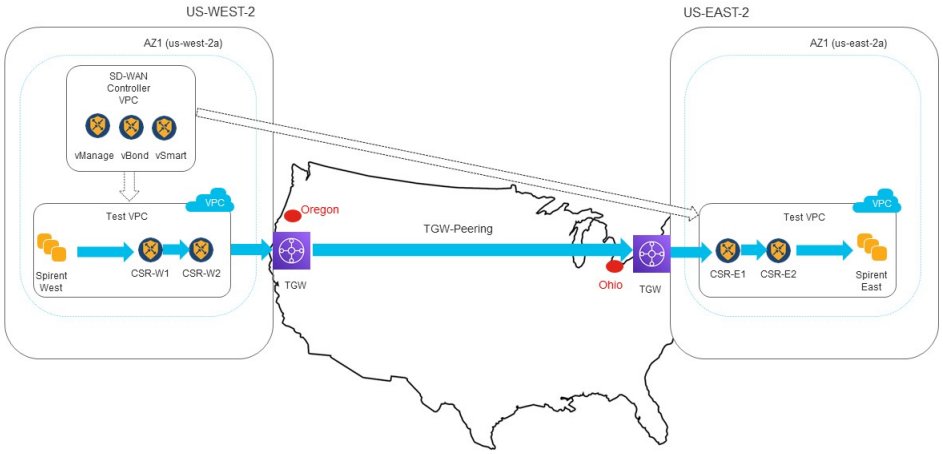


Figure 1: Oregon to Ohio deployment

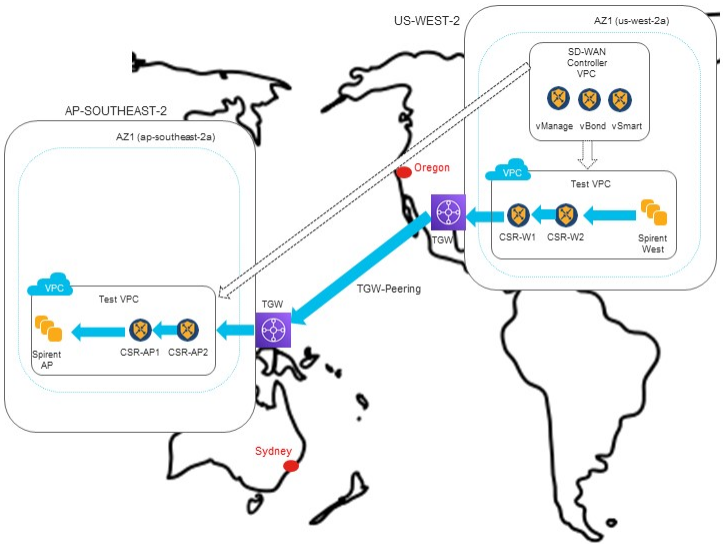


Figure 2: Oregon to Sydney deployment

security can be guaranteed through appropriate security groups that can be expressed through network Access Control Lists (ACL).

Taking into consideration the above, in each AWS region under consideration, we create a different VPC, where the VNFs of our SFC are provisioned as EC2 instances and configured to use specific subnets for management, control, and forwarding purposes. Additionally, we add the proper security groups so only the enterprise branch in this region can have access to the particular VPC and to allow communication between different VPCs (enterprise branches). Fig. 3, illustrates the configuration made in our paper. In particular, we have five subnets. The first subnet (i.e., vpn512) is the management subnet that targets the Internet Gateway (IGW) of the VPC and allows remote access to our instances inside the VPC. The second subnet (i.e., vpn20) is the traffic's generator subnet that sends traffic to the first VNF of the SFC. The third subnet (i.e., vpn10) is the subnet that connects the first with the second VNF. The fourth subnet (i.e. vpn60) sends the traffic from the second VNF to the third VNF, that belongs to a different region and VPC through AWS TGW. Finally, the fifth subnet (i.e. vpn0) is used to connect the VNFs with the SD-WAN controller. The same subnet logic, but with a different /16 CIDR, is applied to the rest of the VPCs in the three AWS regions under consideration.

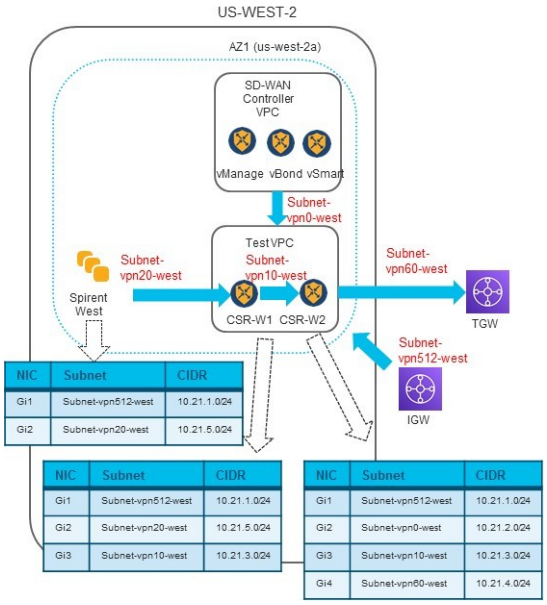


Figure 3: VPC Subnet configuration

3.2.2 Transit Gateway: As mentioned in Section 1 and to remove any performance uncertainties that the Public Internet can introduce, we resort to the AWS TGW solution. AWS TGW allows the interconnection of VPCs through the AWS global network without traveling over the Public internet [4]. AWS global infrastructure covers 26 regions, offering significant benefits in terms of security, availability and performance [6]. Obviously, this can facilitate the delivering of applications and the interconnection of different enterprise networks across the globe.

In our problem at hand, we attach each VPC at a AWS TGW, and then we peer the two AWS TGW together, as shown in Fig. 4. In particular, one AWS TGW is created in every region that we want to interconnect and then the two AWS TGWs are peered together. Following, each AWS TGW needs to be attached with the proper VPC. For example, VPC West will be attached to AWS TGW West and VPC East to AWS TGW East. Finally, the routing tables need to be updated so the inter-region traffic can be forwarded through the AWS TGWs. The same configuration was followed for both deployments under consideration (i.e. Fig. 1 and 2).

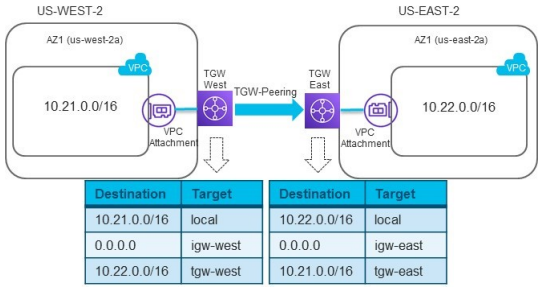


Figure 4: AWS TGW Peering

3.3 Used Tools and Software

To increase the credibility of our work, we used several types of software which are available in AWS marketplace and can be offered to all interested enterprises and stakeholders.

3.3.1 CSR-1000v: The VNFs are deployed using the Cisco Cloud Services Router (CSR) 1000v Series [10]. CSR 1000v is a virtual router, available in AWS, that offers many network features (i.e., IPSec, NAT, DHCP, ACLs, Firewall, etc). For our implementation, we deploy two CSR instances in each region to create the 4-VNF SFC. The first VNF is configured as a virtual router with basic security features (such as Firewall), while the second VNF provides SD-WAN services with encryption through IPSec. The same VNFs in the opposite order are offered in the second region of the deployment. Taking as an example Fig. 1, CSR-W1 is configured as a virtual router, CSR-W2 as IPSec, CSR-E1 as IPSec and CSR-E2 as virtual router. The same configuration is followed, for the second deployment.

3.3.2 SD-WAN:. For the SD-WAN, we use a Cisco SD-WAN solution [11], which comprises of three centralized controllers (vManage, vBond, and vSmart) that have the global overview of the WAN infrastructure. The vSmart controller is responsible for the control plane of the SD-WAN and the vBond is in charge of initial onboarding and authenticating the SD-WAN enabled devices (i.e. CSR1000v in our case). Finally, the vManage is responsible for device configuration via templates using appropriate APIs or Graphical User Interface (GUI). All the controllers can help in exchanging and installing certificates with the CSR1000v virtual routers, while pushing and propagating the appropriate network policies. As shown in Figs. 1 and 2, we deploy a separate VPC for the controllers in Oregon. In reality, there is no location restriction of where to place the controllers, however, we selected Oregon since it is present in both deployments under consideration.

3.3.3 Spirent: For the traffic generator we chose Spirent, as it offers several testing solutions to analyze the performance of NFV platforms including SD-WAN in the Cloud [31]. For our methodology, a Spirent instance is configured in each VPC. The instance in Oregon is configured as the sender, the instance in Ohio as the receiver for the first deployment, and the instance in Sydney as the receiver for the second one. More information regarding the traffic configuration of the generator is provided in subsection 3.4.

3.3.4 Terraform configuration: To automate the configuration of the AWS deployment the Terraform orchestration tool was used [18]. Terraform is an open-source infrastructure as code tool that can also interact with the AWS provider. In particular, through a Terraform script all the necessary AWS components, described in this section can be automatically provisioned through appropriate API requests. This facilitates the automation of the deployment and its repeatability in various AWS regions. Our Terraform script can be found online to allow the easy recreation of our environment for all interested parties [20].

3.4 Measurement Methodology

One of the goals of this analysis is to evaluate the performance between different inter-DC pairs and various time occasions. For the first, we use the two deployments as shown in Figs. 1 and 2. For the

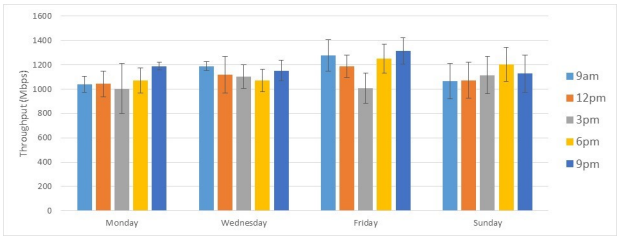
Table 2: Flow Configuration

Flow #	IPv4.Src	Src Port	IPv4.Dest.	Dest. Port
1	10.21.5.32	10000	10.22.5.204	10500
2	10.21.5.33	10001	10.22.5.205	10501
3	10.21.5.34	10002	10.22.5.206	10502
4	10.21.5.35	10003	10.22.5.207	10503
5	10.21.5.36	10004	10.22.5.208	10504
...	...	...	...	...
96	10.21.5.32	10095	10.22.5.204	10595
97	10.21.5.33	10096	10.22.5.205	10596
98	10.21.5.34	10097	10.22.5.206	10597
99	10.21.5.35	10098	10.22.5.207	10598
100	10.21.5.36	10099	10.22.5.208	10599

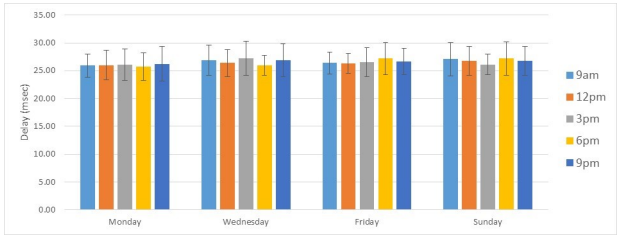
latter, we repeat the experiments 5 times a day and 4 times a week. In particular, we generate traffic between the two regions under test every 3 hours from 9am till 9pm at night (Pacific Standard Time - PST). The experiments are repeated on Mondays, Wednesdays, Fridays, and Sundays to account possible performance differences between weekdays and between weekdays and weekends. Regarding traffic, UDP over IPv4 is considered. Specifically, we create 100 different flows. The flows are identified by the following four-tuple: <source IP, source port, destination IP, destination port>. A breakdown of the flow configuration is provided in Table 2.

For the packet profile, we have configured Spirent to send IP traffic with an Internet Mix (IMIX) or 361 B packet size. IMIX reflects typical traffic that may be found in real communications over the Internet. The reason is that testing with a constant packet size may provide a performance that is not indicative of an operational deployment [25]. IMIX sends IP packets with the following distribution: 7 packets of 40 B size, 4 packets of 576 B, and 1 packet of 1500 B size. However, we noticed that with the particular IMIX profile the performance achieved over AWS was very unstable. Further investigation showed that AWS network infrastructure was rejecting the 40 B and 1500 B IP packets. Thus, after appropriate fine-tuning we customized the IMIX profile as follows: 48 B (IP) / 66 B (Ethernet), 576 B (IP) / 594 B (Ethernet) and 1438 B (IP) / 1456 B (Ethernet). Additionally, we have selected to compare the IMIX profile with the 361 B packet size, which is the average value of the default IMIX packet distribution.

Finally, we follow the RFC2544 [9] testing methodology for extracting the results. In particular, RFC2544 follows a binary search in the range of throughput percentage that can be achieved, i.e., 0-100%, where 100% corresponds to the maximum throughput. In our case, the maximum throughput is 10 Gbps that is limited by the licences used in Spirent and the network capacity of the EC2 instances. The binary search is performed with a resolution of 0.01% while accepting a frame loss rate (FLR) of 0.1%. This means that RFC2544 will try several throughput percentages until it finds the maximum throughput that can meet the targeted FLR. Additionally, according to the requirements of the RFC, each trial should last 60 s to have reliable results. Thus, each measurement lasts about 20 min. This comes in contrast with the pertinent literature that test the performance achieved in a duration of few seconds (i.e. [21, 24]).



(a) Average Throughput



(b) Average Delay

Figure 5: Oregon to Ohio deployment with c5.large/c5.2xlarge instances and IMIX profile

4 EXPERIMENT RESULTS

We ran the experiments over a period of 8 months. In order not to create any bias, we avoided testing the performance during holidays and special days of the year. In total, four experiments were executed, each lasting approximately 2 months (4 weeks for the IMIX profile and 4 weeks for the 361 B packet size). In each experiment we change the pair of regions and the size of the EC2 instances. For the EC2 instances, we used a relatively small VM with 2 CPU cores and 4 GB of RAM (c5.large) and a larger one with 16 CPU cores and 32 GB of RAM (c5.4xlarge). It should be noted that c5.large only comes with 3 Network Interface Cards (NICs). However, the CSRs that are connected with the SD-WAN controllers (herein called CSR 1000v SD-WAN) need 4 NICs (see CSR-W2 in Fig. 3). This is why for the CSRs SD-WAN we selected a bigger instance, namely c5.2xlarge that comes with 4 NICs. Nonetheless, the particular selection should not have any impact on the performance, since the lower size instance will create any performance bottlenecks. The details of the experimentation setup are shown in Table 3.

4.1 Experiment 1

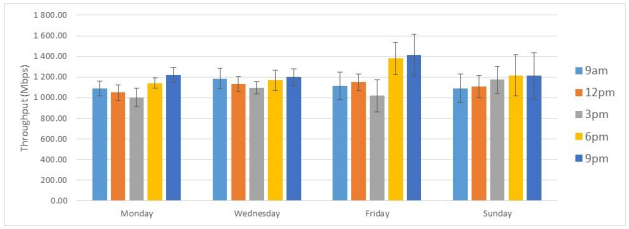
In the first experiment, the SFC is deployed across the US, while using the c5.large and c5.2xlarge instances for each pair of VNFs respectively. Fig. 5 shows the average throughput and delay achieved over a period of a month, while the error bars represent the standard deviation. As can be seen in Fig. 5a, the throughput ranges from approximately 1 Gbps to 1.3 Gbps. What is interesting though, is that the throughput presents the same pattern during the weekdays. Specifically, the throughput starts decreasing after 9am and increases after 6pm and presents the highest value at night. The worst performance is achieved at 3pm with a high standard deviation. Additionally, the highest throughput is achieved on Fridays. Oddly, the performance during the weekend does not seem to be significantly affected by the time period of the day. However, the

Table 3: Instances Configuration and Pricing

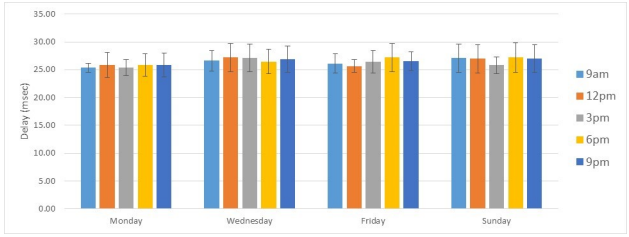
Exp.	VM Type	Region	Instance	CPU	RAM (GB)	Bandwidth (Gbps)	Pricing ($/h)
1^{st}	CSR1000v SD-WAN	Ohio	c5.2xlarge	8	16	Up to 10	0.34
	CSR1000v	Ohio	c5.large	2	4	Up to 10	0.085
2^{nd}	CSR1000v/CSR1000v SD-WAN	Ohio	c5.4xlarge	16	32	Up to 10	0.68
3^{rd}	CSR1000v SD-WAN	Sydney	c5.2xlarge	8	16	Up to 10	0.444
	CSR1000v	Sydney	c5.large	2	4	Up to 10	0.111
4^{th}	CSR1000v/CSR1000v SD-WAN	Sydney	c5.4xlarge	16	32	Up to 10	0.888
1^{st} & 3^{rd}	CSR1000v SD-WAN	Oregon	c5.2xlarge	8	16	Up to 10	0.34
	CSR1000v	Oregon	c5.large	2	4	Up to 10	0.085
2^{nd} & 4^{th}	CSR1000v/CSR1000v SD-WAN	Oregon	c5.4xlarge	16	32	Up to 10	0.68
1^{st} - 4^{th}	Spirent	Ohio	c5.xlarge	4	8	Up to 10	0.17
	Spirent	Sydney	c5.xlarge	4	8	Up to 10	0.222
	Spirent	Oregon	c5.xlarge	4	8	Up to 10	0.17
	vManage	Oregon	c5.4xlarge	16	32	Up to 10	0.68
	vBond & vSmart	Oregon	t2.medium	2	4	Low to Medium	0.046

throughput obtained is not as high as someone would expect. This could indicate of a throttling policy that may be imposed by AWS of how the computational resources are leveraged during the weekends. Regarding, the delay, as shown in Fig. 5b, we notice a very stable behavior. Specifically, the end-to-end delay ranges from 25 to 27 ms, with a standard variation of less than 2 ms for most of the trials. This can be attributed to the AWS TGW peering performance that shows relative stable communication conditions.

Fig. 6 illustrates the performance achieved with 361 B packet size profile. Our first observation is that the same pattern as the IMIX profile is noticed. For instance, as demonstrated in Fig. 6a the throughput drops during the afternoon hours and increases at night. The highest throughput is observed on Fridays. Sunday does not follow the whipsaw behavior of weekdays, but rather presents a relatively stable throughput with a tendency to increase during the late hours of the day. A secondary observation is that the fixed packet size results in a lower standard deviation especially for the first days of the week, while the throughput appears to be slightly higher in comparison with the IMIX profile, ranging from 1 Gbps to 1.4 Gbps. With regards to delay (Fig. 6b), we see an akin conduct with an average value of 25 to 27 ms and a standard deviation of approximately 2 ms. Thus, AWS TGW peering can behave similarly with different packet profiles but with similar average packet size.



(a) Average Throughput



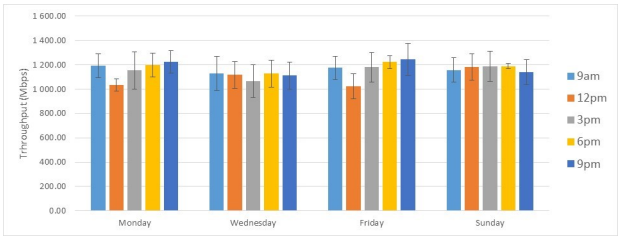
(b) Average Delay

Figure 6: Oregon to Ohio deployment with c5.large/c5.2xlarge instances and 361 B packet size

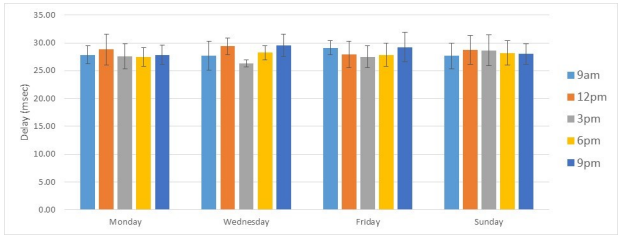
4.2 Experiment 2

Continuing with our experimentation, we keep the US deployment, but the size of the EC2 instances are changed to c5.4xlarge. Fig. 7 illustrates the new attained results for using an IMIX packet profile. Comparing the throughput values between Fig. 5a and Fig. 7a, we observe that the average throughput is increased for the second experiment. This is reasonable, since the resources were practically increased eightfold. However, this improvement is marginal compared to the resource surge. In particular, the overall throughput increase is only of around 25 Mbps. Analyzing, the throughput performance in all of the links of our communication, we found that packets were dropped in AWS TGW Peering link. This signifies, that there is a bandwidth cap imposed by AWS TGW. More details regarding this limit, are provided in Section 4.5. Additionally, throughput seems to be adversely impacted during the afternoon hours of the weekdays and to be improved at night. As in the first experiment, Friday is the day that the maximum values of throughput are noticed, while for Sunday the impact of the time of the measurement seems to be less significant.

Regarding the delay, a small increase is detected for the second experiment with a value ranging from 27 to 29 ms. This average increase of about 1.5 ms between Fig. 5b and 7b can be attributed to the overall throughput increase, since the VNFs need to process a higher volume of packets. What is interesting though is that the delay now demonstrates a higher fluctuation. Taking into consideration that AWS TGW can provide a relatively stable connection, as corroborated in the first experiment, we believe that this variability

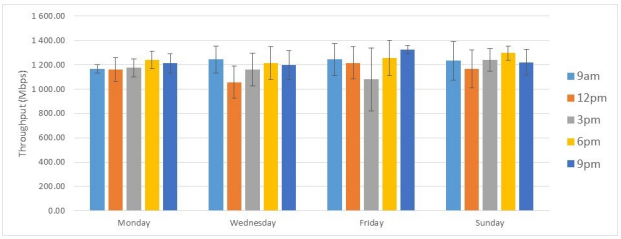


(a) Average Throughput

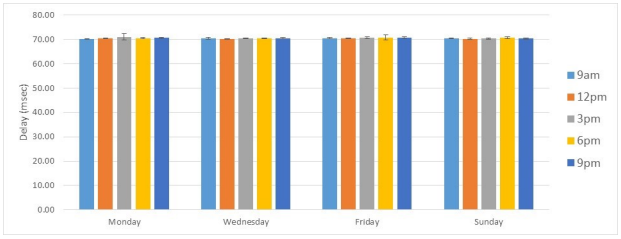


(b) Average Delay

Figure 7: Oregon to Ohio deployment with c5.4xlarge instances and IMIX profile

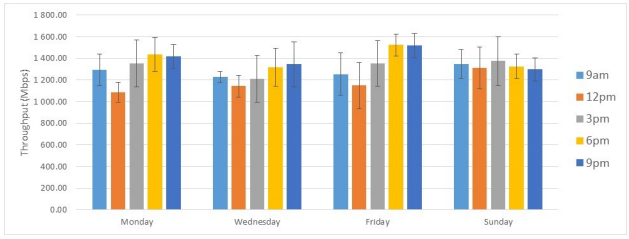


(a) Average Throughput

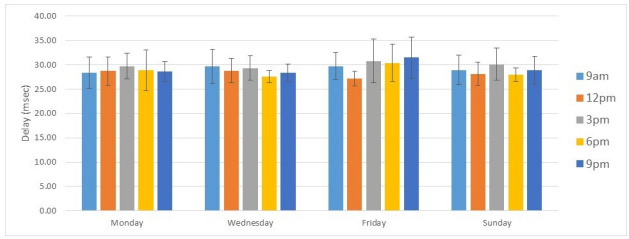


(b) Average Delay

Figure 9: Oregon to Sydney deployment with c5.large/c5.2xlarge instances and IMIX profile



(a) Average Throughput



(b) Average Delay

Figure 8: Oregon to Ohio deployment with c5.4xlarge instances and 361 B packet size

is due to the processing delays involved in the VNFs and less to the propagation delay. Nonetheless, the fluctuation is limited within a margin of less than 3 ms, which is not that significant given that we study a US coast-to-coast connection.

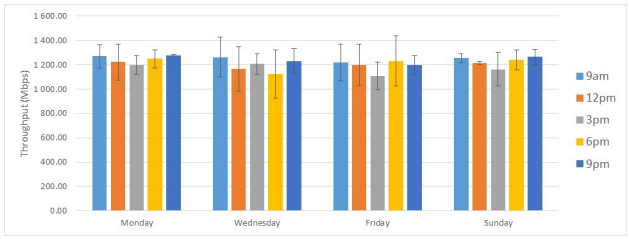
In the second part of this experiment, the packet size is changed to 361 B. As shown in Fig. 8a the throughput is also higher and now ranges between 1.1 Gbps to 1.5 Gbps, with the highest values attained on Friday night. Once more, for all the days of the week there is a tendency for the throughput to increase during the night hours, while for all weekdays the lowest throughput is observed during the afternoon hours. We also notice that the delay in Fig. 8b grows compared to the first experiment. In total, the delay in this part of the experiment is more than 2ms higher than the one in Fig. 6b, which once again can be attributed to the throughput boost. Finally, as in Fig. 7b more fluctuations appear, however the difference between the minimum (Friday at 12pm) and the maximum delay value (Friday at 9pm) is around 4 ms.

4.3 Experiment 3

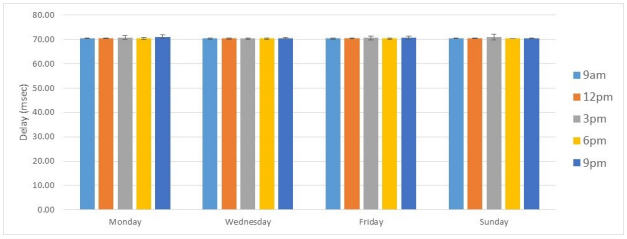
For the third experiment we deploy the SFC as shown in Fig. 2, while the results are shown in Fig. 9. Before analyzing the attained performance, it is worth mentioning that the two regions under consideration present a high time difference of 19 hours. This may affect the behavior that we observed in the two first experiment, where the time difference between Oregon and Ohio is only 3 hours.

Fig. 9a presents the throughput achieved for the particular deployment. Once again, the impact of the time of the measurement is evident for Wednesday and Friday, where the bandwidth is negatively impacted during the afternoon hours. A similar behavior, but not on the same extend, is noticed for Monday and Sunday, which can be attributed to the significant time difference, which can move the bottleneck in the time dimension. The throughput ranges from approximately 1.05 Gbps (on Wednesday at 12pm PST) to 1.3 Gbps (on Friday at 9pm PST), with a total average of 1.204 Gbps.

Regarding delay, as illustrated in Fig. 9b, the performance is significant reduced in comparison with the first experiment (Fig. 6b). This is reasonable, since now the propagation delay is much higher. However, the major observation in this set of experiments is that the delay is very stable while presenting minimal fluctuations. In most of the cases the standard deviation is below 0.20 ms, while the maximum deviation is noticed on Monday at 3pm with a value of 1.3 ms. Hence, as in the first experiment, we can deduce that

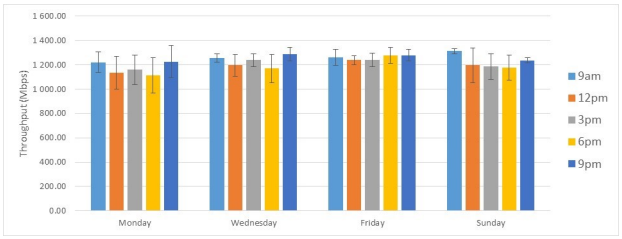


(a) Average Throughput

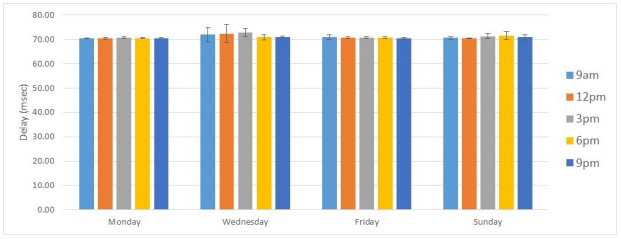


(b) Average Delay

Figure 10: Oregon to Sydney deployment with c5.large/c5.2xlarge instances and 361 B packet size



(a) Average Throughput



(b) Average Delay

Figure 11: Oregon to Sydney deployment with c5.4xlarge instances and IMIX profile

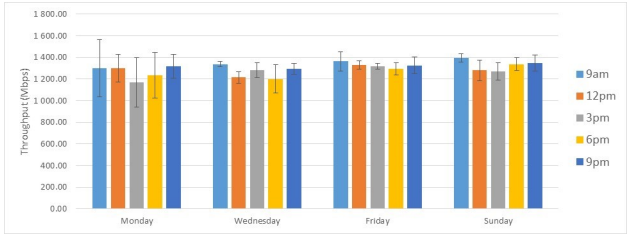
the use of AWS TGW peering can remove any intense fluctuations even for long distance communications.

Similar results are observed when the packet size is fixed at 361 B, as shown in Fig. 10. In terms of throughput (Fig. 10a) and delay (Fig. 10b), approximately the same values are attained as in Fig. 9, while the delay is stable at 70 ms. The performance presents anew the same behavior with lower throughput values during the afternoon hours and higher values at night.

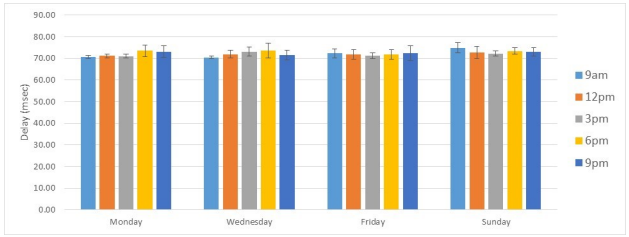
4.4 Experiment 4

In the last experiment, we keep the deployment of Fig. 2 but we increase the size of the VMs to c5.4xlarge. Fig. 11 illustrates the new results for an IMIX packet profile. By comparing Fig. 9 to 11, we observe that the impact of the instance size is marginal, with an increase of less than just 20 Mbps. Once more, AWS TGW was proved to throttle the overall throughput. The adverse impact of the afternoon hours in our measurements persists especially for the first two days of our experimentation. Additionally, the delay in Fig. 11b is also slightly increased in comparison with the third experiment (Fig. 9b). There are more fluctuations in the delay with a standard deviation between 0.1 to 3 ms. However, taking into consideration that the average delay is 71 ms most of the times the standard deviation is negligible.

As a final part of this experiment, the packet size is replaced with a fixed size of 361 B. The juxtaposition of Figs. 12a and 10a reveals again that the size of the instance has a minimal effect on the throughput, which however continues to show the same trend as in all other experiments. The same deductions can be produced by comparing the delay in Figs. 12b and 10b. The delay appears to be slightly increased for a larger EC2 instance, due to the throughput increase. Furthermore, as in the first SFC deployment, the larger instance size creates higher delay fluctuation without though posing any significant alerts regarding performance drifts.



(a) Average Throughput



(b) Average Delay

Figure 12: Oregon to Sydney deployment with c5.4xlarge instances and 361 B packet size

4.5 Discussion

Herein, we try to initiate a discussion regarding the advantages and disadvantages of the deployments under consideration. Table 4 provides the average values of all the experiments. It is noted that each value in the Table corresponds to the average of 80 measurements (5 times a day, 4 times a week, for four weeks). Ideally, we would like to perform a more extensive experimentation, however time and cost constraints did not allow us to get more measurements. Nonetheless, we believe that the 8-month experimentation was enough to get a concrete intuition of the performance behavior.

Table 4: Summary of the results

Experiment	Packet Profile	Thr. (Mbps)	Delay (ms)
1	IMIX	1129.94	26.43
1	361 B	1158.49	26.44
2	IMIX	1153.05	28.20
2	361 B	1314.42	29.08
3	IMIX	1204.83	70.51
3	361 B	1214.79	70.52
4	IMIX	1220.44	71.20
4	361 B	1295.67	72.30

Analyzing the results in Table 4, the following three major observations are inferred. Firstly, for all the experiments the use of a fixed packet size of 361 B instead of an IMIX profile leads to a slight throughput and delay increase. The reason for this behavior may be attributed to the fact that the 361 B is the average value of the default IMIX profile. However, as explained in Section 3 we had to moderately change the packet sizes of the IMIX profile that gives an average value of 357 B. Another reason, is that the majority of the packets in the IMIX case are small packets of around 66 B, which may be processed much faster than a fixed packet size of 361 B.

Secondly, the impact of increasing the instance size is insignificant. As stated in Sections 4.2 and 4.4 the bottleneck was found to lie in the AWS TGW Peering link, where we noticed a considerable packet drop rate. This led us to search if there is a known limitation imposed by AWS TGW, since from separate testing, we found that the c5.4xlarge instance of CSR 1000v SD-WAN could achieve up to 2.2 Gbps. Indeed, our findings were corroborated, as AWS TGW limits the bandwidth to around 1.25 Gbps per VPN connection [7]. In practice, we have one IPSec tunnel (thus, one VPN connection), and this is why our measured bandwidth is limited between 1.2 to 1.3 Gbps for the c5.4xlarge instance, as shown in Table 4.

Lastly, the delay change for the different scenarios is minimum. For the first deployment the maximum difference is between the second experiment when using 361 B and the first experiment when using IMIX. This difference is only of 2.65 ms. For the second deployment the delay difference is even less with a value of 1.79 ms for the different packet size and instance settings. Inevitably, this can be attributed to the robust performance of AWS TGW.

Regarding the cost of the deployment, Table 3 shows the pricing of the various EC2 instances used in ($/h). The prices found in AWS Marketplace for the selected VM Types follow the same model as the typical Linux instances of AWS. The main observation from this pricing list is that when changing from a relatively small instance to a large instance, the resources are eight times more and the price is quadrupled. Obviously, it is not reasonable to expect an analogous performance increase for the VNFs [12]. However, as explained above the bottleneck is generated by AWS TGW and not the size of the instance. One way to alleviate this, is to use multiple VPN tunnels and balance the load between them, as already investigated by AWS and Cisco [19], and thus we decided not to reproduce. Another solution, could be to switch from IPSec to a GRE tunnel. However, this can raise security issues and cannot be comparable with our proposed deployment due to the transport mechanism

Table 5: AWS TGW Pricing

Region	Price per Attachment ($/h)	Price per GB ($)
Ohio	0.05	0.02
Oregon	0.05	0.02
Sydney	0.07	0.02

change. Thus, when planning the resources, it is of utmost importance to know the throughput requirements. Similarly, it is equally important to have insights on which part of the communication will cause first any performance bottlenecks, and thus will regulate the maximum throughput. Following, the resources allocated to the VNFs can be accordingly adjusted to match that performance. This can lead to significant deployment cost reductions in the long-term without wasting resources or money.

Finally, we would like to analyze the trade-off between performance and cost when using the AWS TGW peering. Table 5 illustrates the pricing of the AWS TGW connections according to the region of the attachment [5]. As can been seen, AWS TGW includes two prices, the price per attachment and the price to send data through an attachment. For example, in our case if we want to send 1 GB of data between Oregon and Ohio, a charge of 0.02 $ will be added for the data processing involved in Oregon and an additional 0.02 $ in Ohio. Yet, this additional cost can be counterbalanced by the attained delay performance. As shown in the previous subsections the delay for the first deployment is around 27 ms and for the second 71 ms. The corresponding delays when using the Public Internet are around 70 ms and 176 ms respectively [14]. Especially, for the transpacific, the 70 ms performance is even better than very expensive MPLS services currently available [33].

Moreover, even when connecting the AWS regions through Public Internet we have to take into account the additional charges for sending traffic from/to AWS and the higher fluctuations noticed from an often intermittent connection [3]. Thus, there is a need to understand the requirements of the application in terms of delay and robustness and perform a cost analysis to take the best decision.

5 CONCLUSION

In this paper, we have analyzed for the first time the deployment of an SD-WAN controlled SFC over an inter-DC communication, using the AWS infrastructure. In particular, we have considered two inter-region deployment scenarios. For both scenarios a SFC with 4 VNFs is deployed offering routing and encryption services. Additionally, the AWS TGW component was used to interconnect the regions under consideration. An extended and realistic experimentation was made spanning over a period of 8 months, testing different packet profiles, different size of instances, and analyzing the impact of time and space in the attained performance. The results revealed, that the throughput performance can present noticeable fluctuations in accordance to the time of the measurement. Finally, the use of AWS TGW guarantee a robust performance in terms of the delay, however limiting the maximum bandwidth achieved. As a future work we aim to deploy SFCs spanning across more than two regions and repeat the experiments on different CSPs, such as Azure and Google Cloud.

REFERENCES

[1] AWS. 2022. Amazon Virtual Private Cloud (Amazon VPC). https://aws.amazon.com/vpc/.

[2] AWS. 2022. Amazon VPC FAQs. https://aws.amazon.com/vpc/faqs/.

[3] AWS. 2022. AWS Pricing Calculator. https://calculator.aws/#/.

[4] AWS. 2022. AWS Transit Gateway. https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html.

[5] AWS. 2022. AWS Transit Gateway Pricing. https://aws.amazon.com/transit-gateway/pricing/.

[6] AWS. 2022. Global Infrastructure. https://aws.amazon.com/about-aws/global-infrastructure/.

[7] AWS. 2022. Quotas for your transit gateways. https://docs.aws.amazon.com/vpc/latest/tgw/transit-gateway-quotas.html.

[8] Asma Ben Hamed, Aris Leivadeas, Matthias Falkner, and Nikolai Pitaev. 2020. VNF Chaining Performance Characterization under Multi-Feature and Oversubscription Using SR-IOV. *Informatics* 7, 3 (2020). https://doi.org/10.3390/informatics7030033

[9] S. Bradner and J. McQuaid. 1999. RFC2544: Benchmarking Methodology for Network Interconnect Devices.

[10] Cisco. 2020. Cisco Cloud Services Router 1000v Data Sheet. https://www.cisco.com/c/en/us/products/collateral/routers/cloud-services-router-1000v-series/data_sheet-c78-733443.html.

[11] Cisco. 2022. Cisco SD-WAN Getting Started Guide. https://www.cisco.com/c/en/us/td/docs/routers/sdwan/configuration/sdwan-xe-gs-book/system-overview.html.

[12] Matthias Falkner, Aris Leivadeas, Ioannis Lambadaris, and George Kesidis. 2016. Performance analysis of virtualized network functions on virtualized systems architectures. In *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*. 71–76. https://doi.org/10.1109/CAMAD.2016.7790333

[13] Marco Forconesi, Gustavo Sutter, Sergio Lopez-Buedo, Jorge E. Lopez de Vergara, and Javier Aracil. 2014. Bridging the gap between hardware and software open source network developments. *IEEE Network* 28, 5 (2014), 13–19. https://doi.org/10.1109/MNET.2014.6915434

[14] Anshul Gandhi and Justin Chan. 2015. Analyzing the Network for AWS Distributed Cloud Computing. *SIGMETRICS Perform. Eval. Rev.* 43, 3 (nov 2015), 12–15. https://doi.org/10.1145/2847220.2847224

[15] José Luis García-Dorado and Sanjay G. Rao. 2019. Cost-aware Multi Data-Center Bulk Transfers in the Cloud from a Customer-Side Perspective. *IEEE Transactions on Cloud Computing* 7, 1 (2019), 34–47. https://doi.org/10.1109/TCC.2015.2469666

[16] Nadir Ghrada, Mohamed Faten Zhani, and Yehia Elkhatib. 2018. Price and Performance of Cloud-hosted Virtual Network Functions: Analysis and Future Challenges. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. 482–487. https://doi.org/10.1109/NETSOFT.2018.8460032

[17] Mohammad Hajjat, Ruiqi Liu, Yiyang Chang, T. S. Eugene Ng, and Sanjay Rao. 2015. Application-specific configuration selection in the cloud: Impact of provider policy and potential of systematic testing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. 873–881. https://doi.org/10.1109/INFOCOM.2015.7218458

[18] Hashicorp. 2022. Terraform. https://www.terraform.io/.

[19] Vinod Kataria and Sreekanth Krishnavajjala. 2020. Scaling VPN throughput using AWS Transit Gateway. https://aws.amazon.com/blogs/networking-and-content-delivery/scaling-vpn-throughput-using-aws-transit-gateway/.

[20] A. Leivadeas. 2022. AWS Terraform Configuration. https://github.com/arisleiv/AWS_Terraform.

[21] Aris Leivadeas, Matthias Falkner, Ioannis Lambadaris, and George Kesidis. 2016. Dynamic traffic steering of multi-tenant virtualized network functions in SDN enabled data centers. In *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*. 65–70. https://doi.org/10.1109/CAMAD.2016.7790332

[22] Aris Leivadeas, Matthias Falkner, Ioannis Lambadaris, and George Kesidis. 2017. Optimal virtualized network function allocation for an SDN enabled cloud. *Computer Standards & Interfaces* 54 (2017), 266–278. https://doi.org/10.1016/j.csi.2017.01.001 SI: Standardization SDN&NFV.

[23] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. 2010. CloudCmp: Comparing Public Cloud Providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (Melbourne, Australia) *(IMC '10)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/1879141.1879143

[24] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2016. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials* 18, 1 (2016), 236–262. https://doi.org/10.1109/COMST.2015.2477041

[25] A. Morton. 2013. RFC6895: IMIX Genome: Specification of Variable Packet Sizes for Additional Testing.

[26] Fabio Palumbo, Giuseppe Aceto, Alessio Botta, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapé. 2021. Characterization and analysis of cloud-to-user latency: The case of Azure and AWS. *Computer Networks* 184 (2021), 107693. https://doi.org/10.1016/j.comnet.2020.107693

[27] Valerio Persico, Alessio Botta, Pietro Marchetta, Antonio Montieri, and Antonio Pescap. 2017. On the Performance of the Wide-Area Networks Interconnecting Public-Cloud Datacenters around the Globe. *Comput. Netw.* 112, C (jan 2017), 67–83. https://doi.org/10.1016/j.comnet.2016.10.013

[28] Nikolai Pitaev, Matthias Falkner, Aris Leivadeas, and Ioannis Lambadaris. 2018. Characterizing the Performance of Concurrent Virtualized Network Functions with OVS-DPDK, FD.IO VPP and SR-IOV. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering* (Berlin, Germany) *(ICPE '18)*. Association for Computing Machinery, New York, NY, USA, 285–292. https://doi.org/10.1145/3184407.3184437

[29] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. 2010. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *Proc. VLDB Endow.* 3, 1–2 (sep 2010), 460–471. https://doi.org/10.14778/1920841.1920902

[30] Francesco Spinelli, Luigi Iannone, and Jerome Tollet. 2019. Chaining your Virtual Private Clouds with Segment Routing. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 1027–1028. https://doi.org/10.1109/INFCOMW.2019.8845113

[31] Spirent. 2022. Cloud Testing. https://www.spirent.com/products/cloud-infrastructure-test.

[32] L. S. Vailshery. 2022. Current enterprise public cloud adoption worldwide from 2017 to 2020. https://www.statista.com/statistics/511508/worldwide-survey-public-coud-services-running-applications-enterprises/.

[33] Verizon. 2022. IP Latency Statistics. https://www.verizon.com/business/terms/latency/.

[34] Zhenjie Yang, Yong Cui, Baochun Li, Yadong Liu, and Yi Xu. 2019. Software-Defined Wide Area Network (SD-WAN): Architecture, Advances and Opportunities. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. 1–9. https://doi.org/10.1109/ICCCN.2019.8847124

[35] Bahador Yeganeh, Ramakrishnan Durairajan, Reza Rejaie, and Walter Willinger. 2020. A First Comparative Characterization of Multi-cloud Connectivity in Today's Internet. In *Passive and Active Measurement*, Anna Sperotto, Alberto Dainotti, and Burkhard Stiller (Eds.). Springer International Publishing, Cham, 193–210.