# [DRAFT] Evaluation of Network Expansion on OpenStack using VPNaaS

Ansh Sarkar, Pranav Bhatt
*4th semester B.Tech in CSE*
*PES University*
Bangalore, India
anshsarkar1@gmail.com, adpranavb2000@gmail.com

Prof. Dinkar Sitaram, Assistant Prof. Usha Devi
*Centre for Cloud Computing and Big Data*
*PES University*
Bangalore, India
dinkars@pes,edu, ushadevibg@pes.edu

***Abstract*—Abstract:- OpenStack is a platform that enables users to test and deploy SDDCs (Software-Defined Data Centers) and custom cloud deployments as per the users needs. It virtualizes all aspects of a server; storage, memory, processing, and the focus of our project today, networking. It does this by combining various services handling a specific part of the hardware software stack. The networking is handled by the service "Neutron", which utilizes OpenVN as the primary back-end. A major paradigm being developed within OpenStack is federation of clouds, which is combining various parts of said virtualized hardware software in order to efficiently and securely scale any cloud operations. This paper explores the VPNaaS plugin which is a step towards realising federation within OpenStack, the features it provides, its performance, and how various encryption standards affect its operation, and its use cases.**

***Index Terms*—OpenStack, Neutron, VPNaaS, Benchmarking, testing**

## I. Introduction

To conduct tests, an IPSec tunnel using the VPNaaS plugin was created in order to connect two different virtual networks. The network performance was then evaluated with and without the IPSec tunnel, as well as test how the performance was affected by different encryption standards. The VPNaaS plugin is provided by the OpenStack service Neutron. For the VPN connection to be successfully activated, parameters certain parameters such as IKE policy, IPSec policy and Endpoint groups need to be predefined in order for VPNaaS to establish a site-to-site connection. Other ways to connect two instances on different networks include GRE tunneling, associating floating IPs or creating a linux/OVS bridge. However these do not provide an encrypted tunnel without modifications.

Our tests involve comparing the performance of the VPNaaS plugin against the performance of a connection without encryption, and how efficiently it handles various levels of encryption. As seen with other forms of VPN tunnels, due to VPNaaS performing encryption and decryption of packets at the routers, the network performance decreases. The IKE policy and IPSec policy can be configured for different levels of encryptions such that one could prioritise either performance or security according to their needs.

The latest version of DevStack was used to set up all the necessary services to run on one system. DevStack is a series of extensible scripts used to quickly bring up a complete OpenStack environment based on the available distribution. The configuration file used to provide parameters for the install is set up such that VPNaaS plugin will be active for both Neutron and Horizon services. On the successful installation of DevStack, using Neutron we create two networks and routers. The subnets and interfaces are accordingly added and then two ubuntu instances are created and attached to these two networks. IKE policy and IPSec policy is configured to establish an IPSec tunnel between these 2 instances.

## II. Technologies and Concepts used

The entire implementation of setting up Openstack and configuring the networks and VPN was carried out on Google cloud platform (GCP) instances. Google cloud platform provides easy access to cloud systems and other computing services. GCP is only a medium for the deployment of our project that majorly deals with the concepts of VPNaaS and Neutron services provided by Openstack.

### A. *OpenStack*

OpenStack is a set of software tools for building and managing cloud computing platforms for public and personal clouds. OpenStack lets users deploy virtual machines and other instances that handle different tasks for managing a cloud environment on the fly which can be customized by the user according to his needs by adding or removing services. It makes horizontal scaling easy, which suggests that the tasks that benefit from running concurrently can easily serve more or fewer users on the fly by just spinning up more instances.

Openstack can be deployed in multiple nodes or all in one configuration. Multiple node architecture has a group of services working in a single host system. A number of host machines are set up with services and then configured to communicate between them.

For our implementation, an all in one set up was the best choice as it provides each and every service in a single host.

To deploy this installation of OpenStack we use DevStack. DevStack is a series of extensible scripts used to quickly bring up a complete OpenStack environment based on the latest versions of everything from git master. It is used interactively as a development environment and as the basis for much of the OpenStack project's functional testing. DevStack uses just a configuration file and other dependencies that have to be manually configured to successfully deploy the all in one installation of OpenStack.

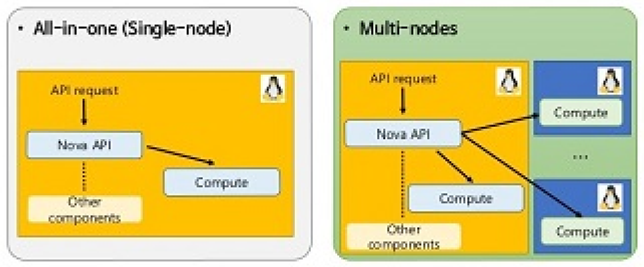

Fig. 1. OpenStack single node and multi node deployment.

## B. Neutron

Neutron is an OpenStack project that virtualizes networking, and provides its services to other interfaces in OpenStack.

Neutron allows creating virtual networks, subnets, and routers. These virtual objects mimic their physical counterparts: networks contain subnets, and routers route traffic between subnets and networks.

There is at least one external network for any given Networking set up. The external network represents a view into a slice of the physical, external network accessible outside the OpenStack installation. This is achieved by OpenVN creating an external bridge. IP addresses on the external network are accessible by anybody physically on the outside network. In addition to external networks, a Neutron set up could have one or more internal networks. These software-defined networks connect directly to the VMs. Only the VMs on any given internal network, or those on subnets connected through interfaces to a similar router, can access other VMs connected to that network directly.

For the outside network to access VMs, and vice versa, routers between the networks are needed. In such cases, each router must have one gateway connected to an external network and one or more of its interfaces connected to internal networks. Like a physical router, subnets can access machines on other subnets that are connected to the same router, and machines can access the outside network using the gateway for the router.

Neutron also allows the defining of security groups. Security groups enable administrators to define firewall rules in groups. A VM can belong to multiple security groups, and Neutron applies the rules in those security groups to restrict or unblock ports, port ranges, or certain traffic types for that VM.

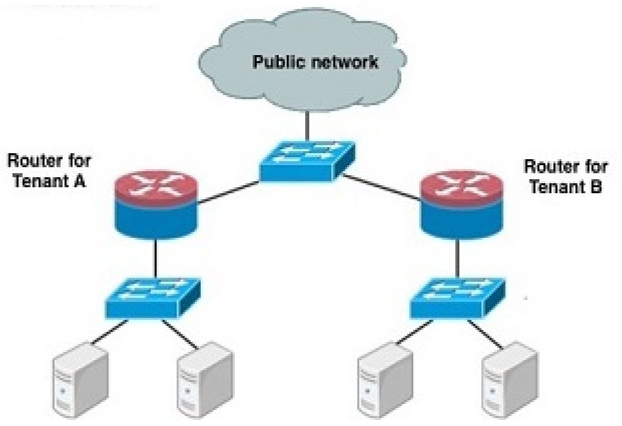

Fig. 2. Neutron example.

## C. VPNaaS

VPNaaS API is maintained as a separate repo and is implemented as an extension to the OpenStack networking API. The VPNaaS API works with the Neutron API simultaneously to provide VPN services for OpenStack.VPNaaS provides Virtual Private Network as a Service (VPNaaS) capabilities to Neutron.

VPNaaS takes advantage of the dependencies such as IKE policy, IPSEC policy to establish a VPN connection through an IPSec tunnel. It protects the data flow between one or more peers. It follows a number of protocols to secure the communications at the network or packet processing layer.

IPsec makes use of tunneling. The IPSec tunnel is created by defining a number of characteristics, parameters, and security measures protocols so that the data packets are transferred securely through the tunnel. IPsec offers numerous technologies and encryption modes that can be configured depending on the use case.The characteristics of this tunnel are defined using the IKE policy and the IPSEC policy. These define the encryption and authentication of the data transfer and peers respectively.

VPNaaS follows five basic steps to implement this tunnel:

- Traffic Initiation
- IKE phase 1
- IKE phase 2
- Data transfer
- Tunnel termination

*1) IKE Policy:* Internet Key Exchange (IKE) is a key management protocol that is used to automatically establish IPsec security associations (SAs), negotiate and distribute IPsec encryption keys, and most importantly to authenticate IPsec peers. The IKE negotiation comprises two phases i.e IKE Phase 1 where it negotiated a security association between two IKE peers, which enables a secure communication between the two peers in Phase 2. During Phase 2 negotiation,it establishes SAs for other applications, such as IPsec. To secure the IKE negotiation between the two peers they use a set of algorithms that are defined in the IKE proposal. IKE negotiation begins by each peer agreeing on a shared IKE policy. This policy contains which security parameters will be used to protect subsequent IKE negotiations.

*2) IPSec Policy:* An IPsec policy defines a mix of security parameters (IPsec proposals) used during IPsec negotiation. It defines Perfect Forward Secrecy (PFS) and therefore the proposals needed for the connection. During the IPsec negotiation, IPsec looks for a proposal that's the identical on both peers. The peer that initiates the negotiation sends all its policies to the remote peer, and therefore the remote peer tries to search out a match. The host checks if the packet should be transmitted using IPsec or not as ese packet traffic triggers the security policy for themselves.This is done when the system sending the packet apply an appropriate encryption. The host also checks the incoming packets if they are properly encrypted or not.

*3) SHA and AES Encryption:* In order to facilitate a general and powerful encryption to be used in government communication, bands and medical establishments that accommodate sensitive data the National Institute of Standards came up with the Advanced Encryption Standards(AES). The encryption method created by AES focuses on safer encryptions by assigning them a singular key. The AES encryptions can not be decrypted unless they need the precise unique key defined within the AES method. These keys are unbreakable and demanding to guess with brute-force approach or any other means. AES represents a secure and secure thanks to hide the information from any unwanted third parties which can not be compromised unless someone has access to the key or breaks the encryption itself.

The SHA or the Secure Hash Standards defines a straightforward standard algorithm for a hash function. A hash function is analogous to an encryption algorithm therein it's meant to scramble data in a way. A hash function performs transformations on the message to generate a value which is called the hash value which is usually smaller than the initial message which might be used for the authentication of the message later. While sending data or any information, the SHA functions are often applied on both the sender and therefore the receiver ends. The output of both the functions is checked and if the hash values produced are not same then the message has been tampered or damaged with. To provide encryption and authentication of sensitive messages the SHA standard algorithm can be coupled with AES encryption.

## III. Setup and Tests performed

The tests that were performed were using the iPerf program which measures data throughput and packet loss, traceroute to measure network latency, and we shall also test a real world scenario of transferring a file of size 3012 MB using SCP, followed by another test transferring the file via HTTP to mitigate the overhead introduced by SCP. We felt the best way to test the VPNaaS plugin's capability was to remove any bottle neck potential caused by the medium itself. We, therefore, limited ourselves to deploying it on a single host, such that all the data transfer would occur within the host itself.

The topology is as shown in Figure 3, which consists of the network branching from a single external network, with an instance attached to it in order for packet inspection. We then have 2 routers attached to the external network, each leading off to their own private networks. Each of these networks have a single instance connected to them for testing the two types of connections. We assign floating IPs to the instances for testing the connection without using VPNaaS, and we use the router endpoints to configure the point-to-point VPN between the two routers, allowing us to use the internal IPs of the instances to test the connection over VPN.

The VPN connections themselves are created using 3 configurations to test how various levels of security would affect performance. The first configuration for the IPSec site-to-site connection represents a connection with minimal security (SHA1, AES128, IKE v1, DH Group 5). The second configuration represents a realistic deployment for this plugin, with a good balance between security and performance (SHA256, AES128, IKE v2, DH Group 14). The third configuration represents "military grade" encryption standards (SHA256, AES256, IKE v2, DH Group 14).

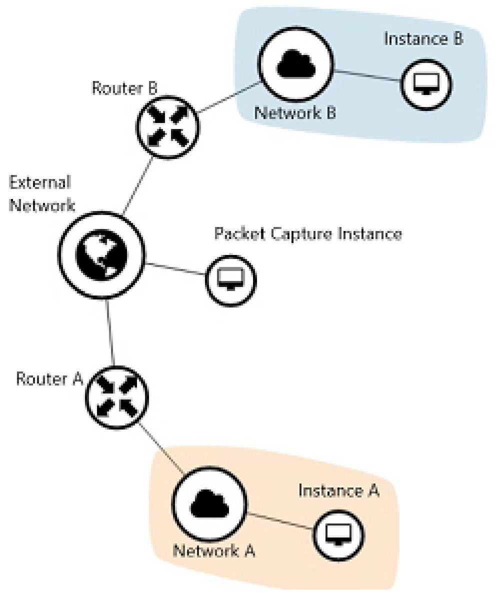

Fig. 3. Topology created for the implementation

## IV. Results

### A. iPerf

After running tests using iPerf on the previously mentioned configurations, the connection speed without VPNaaS averaged 1390 Mbit/s, the VPNaaS connections (SHA1, AES128), (SHA256, AES128), and (SHA256, AES256) averaged speeds of 491 MBit/s, 310 MBit/s, and 270 MBit/s respectively. Figure 4 shows the speeds and the clear trend that emerges.We can see that there is a stark difference in the throughput due to the overhead added by the encryption and decryption process introduced by VPNaaS. We also see how specific to

VPNaaS, the speeds are affected as we move through different standards of encryption, where increasing the SHA bits adds a significantly larger overhead than increasing the AES bits. However, this conclusion could also be the outcome of more time being available to both the stages due to the overhead.

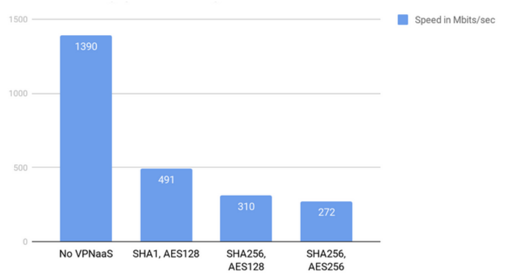

Fig. 4. Network throughput test using iPerf

### B. Traceroute

The latency differences had to be measured on a microse ond scale as the overheads were caused solely due to the lin itations of software virtualisation. In real world applications, the latency overhead due to VPNaaS would be negligible. As shown in Figure 5, the average latency was 1.3194 ms for the scenario without VPNaaS, 2.29 ms with a (SHA1, AES128) configuration for the VPN connection, 2.3106 ms with a (SHA256, AES128) configuration, and 2.6562 ms with a (SHA256, AES256) configuration. The VPNaaS adds approximately latency of 0.97 ms, with minor increases as the level of encryption increases

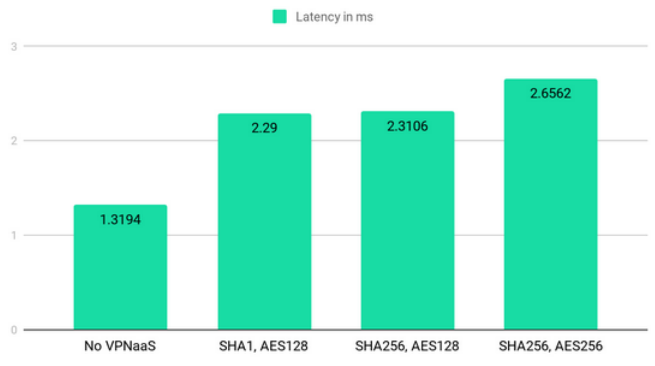

Fig. 5. Latency test using Traceroute

### C. Secure Copy Protocol(SCP)

To test a real world file transfer scenario, we transferred a file of size 512 MB from one instance to another over the various connection scenarios as stated previously. As SCP functions over SSH, the overhead due to the encryption acts as the bottleneck, so the performance we observed wasn't anywhere near the throughput we observed from iPerf. As shown in Figure 6, the average transfer rate was 6.7 MBytes/s for the scenario without VPNaaS, 6.5 MBytes/s with a (SHA1, AES128) configuration for the VPN connection, 6.4 MBytes/s with a (SHA256, AES128) configuration, and 6.5 MBytes/s with a (SHA256, AES256) configuration. These readings are within margin of error, but would represent a realistic workload.

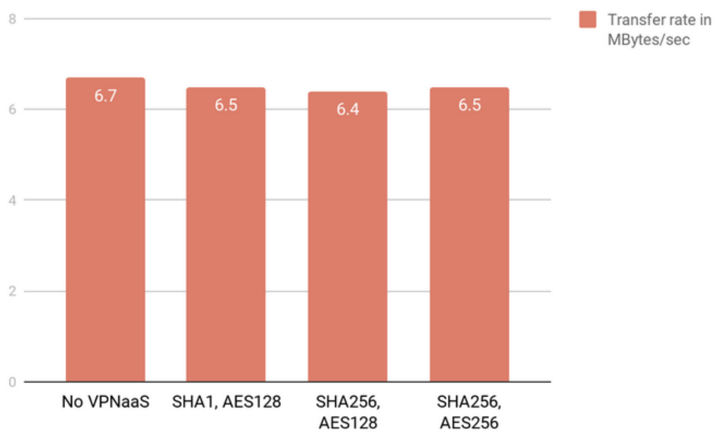

Fig. 6. SCP transfer test

### D. HTTP File Transfer

As the test with SCP was limited by the encryption overhead caused by the underlying SSH tunnel, we conducted another test that could represent real world performance scenarios was a file transfer done using an HTTP server. We used the same 512 MB file for the testing purposes. Although the speeds aren't similar to what we observed from iPerf, this is expected as it only deals with transmitting and receiving the packets to perform its benchmark, and doesn't handle the actual writing and piecing together of the data. As shown in Figure 7, the average transfer rate was 31 MBytes/s for the scenario without VPNaaS, 24.33 MBytes/s with a (SHA1, AES128) configuration for the VPN connection, 22 MBytes/s with a (SHA256, AES128) configuration, and 20 MBytes/s with a (SHA256, AES256) configuration.

## V. Future Work

As shown in Figure 3 the topology clearly shows that both the routers and instances were created on a single OpenStack machine. The networks, routers, subnets and the instances are created virtually on only one GCP instance in order to eliminate the external factors and see how well VPNaaS performs. In future, to test the reliability of VPNaaS the same procedures could be used to establish an IPSec tunnel between two separate OpenStack nodes. A router, instance and network can be set up on both of these OpenStack machines making sure that the two different machines are in the same network or subnet. We could then test how reliable the tunnel is in various connectivity conditions, and also find out if multiple tunnels would affect the performance of VPNaaS. The next stage would be to figure out a way to connect OpenStack

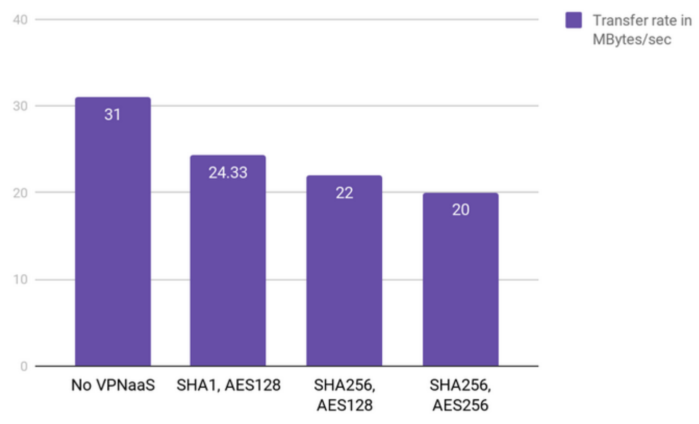

Fig. 7. Points Scored

nodes present in different networks, which would be a stepping stone towards various other topologies and control planes. This particular implementation could be scaled up and a number of instances could be connected with each other.

The main advancement in this case study would be Federating the Neutron service. Modifying VPNaaS could be a way to federate Neutron. It would be one of the most important modifications as enabling such capabilities would vastly extend the usability of the platform.

## VI. Conclusion

VPNaaS is a powerful plugin that allows an IPSec site-to-site connection between two routers. This enables machines on two isolated branches to securely communicate with each other. However, depending on the workload, there may be a performance penalty due to its implementation, and the encryption standard specified by the user. We hope our findings would help a user decide if the plugin is right for their use cases, and inform them of the performance penalty caused due to the encryption standard they select.

## References

[1] M. Kimmerlin, P. Hasselmeyer, S. Heikkila, M. Plauth, P. Parol, and P. Sarolahti, “Network Expansion in OpenStack Cloud Federations,” to appear in Proceedings of 2017 European Conference on Networks and Communications (EuCNC), Jun. 2017.
[2] Layer 3 Networking in Neutron - via Layer 3 agent and OpenVSwitch. [Online]. Available: http://docs.openstack.org/ developer/neutron/devref/layer3.html
[3] OpenStack Foundation. (2017, Jan.) Virtual Private Network as a Service.[Online]. Available: https://wiki.openstack.org/wiki/Neutron/VPNaaS
[4] Luca Tartarini, Marek Denis, ”Federation of OpenStack clouds”
[5] Diana Andreea Popescu, ”Latency-driven performance in data centres”
[6] Mael Kimmerlin , Peer Hasselmeyer, Andreas Ripke, ”Multipath Cloud Federation”
[7] Blog: What Are The Major Challenges Of Adopting A Hybrid Cloud http://www.micoresolutions.com/major-challenges-adopting-hybrid-cloud-approach/
[8] Devstack documentation from https://docs.openstack.org/devstack/latest/