Network Expansion in OpenStack Cloud Federations

Maël Kimmerlin*, Peer Hasselmeyer†, Seppo Heikkilä‡, Max Plauth§, Paweł Parol¶ and Pasi Sarolahti*
*School of Electrical Engineering, Aalto University, Finland
†NEC Laboratories Europe, Heidelberg, Germany
‡Helsinki Institute of Physics, CERN, Geneva, Switzerland
§Hasso Plattner Institute, University of Potsdam, Germany
¶Orange Polska, Poland

***Abstract*—Cloud federation is receiving increasing attention due to the benefits of resilience and locality it brings to cloud providers and users. Our analysis of three diverse use cases shows that existing solutions are not addressing the federation needs of such use case applications. In this paper, we present an alternative approach to network federation, providing a model based on cloud-to-cloud agreements. In our scenarios, companies hosting their own OpenStack clouds need to run machines transparently in another cloud, provided by a company they have an agreement with. Our solution provides multiple benefits to cloud providers and users detailed in this paper. Our implementation outperforms the VPNaaS solution in OpenStack in terms of throughput.**

I. Introduction

Cloud computing has become a widely adopted method of running applications. One of its main benefits is the possibility to acquire computing, storage and networking resources when needed. However, this usually means running the applications in a single location or within a single cloud provider. This comes with a number of potential issues, in particular dependence on single points of failure and latency differences depending on client location. In order to mitigate these two issues, it is advisable to use multiple locations, and multiple cloud providers. Such application distribution solves the problem of a single point of failure as separate data centers usually have independent energy supplies and network uplinks, rendering the scenario of parallel outages of all data center locations and providers very unlikely. With the use of multiple providers, location diversity is increased and it is expected that data centers get closer to the clients on average, reducing average application latency, which is becoming increasingly important with modern cloud workloads.

Interconnecting cloud data centers and extending applications beyond a single cloud provider are not realized easily. This paper identifies the main issues that arise when interconnecting cloud sites from a networking point of view and describes the architectural options for solving such issues. We also report on the solution which the authors have implemented in an OpenStack-based cloud testbed that consists of five sites distributed across Europe. Results of some initial experiments on that testbed are provided to show the efficiency of the solutions proposed.

The main contribution of this paper is the identification of issues to be addressed when interconnecting multiple cloud data centers. Furthermore, solutions addressing these issues are discussed and, finally, a preliminary evaluation of the implemented solutions is provided. The next section introduces the use cases on which we based our design. Section III presents the design of our agent and Section IV provides some performance measurements. The following section compares our approach to existing solutions.

II. Use cases

As a basis for our research, we analyze three applications which are representative of the broad range of cloud uses.

A. Applications

1) Distributed in-memory database: Hyrise is an open source in-memory research database[1]. Hyrise-R, its cloud-based extension, employs lazy master replication, with a master node, several replica nodes and one dispatcher node.

The master is the only node accepting write operations, propagating them to replicas. The dispatcher merely acts as a load-balancer that redirects read operations to all replicated nodes and therefore can speed up the entire system. Spanning replicated setups across federated clouds brings forth several new use cases. First, replication across geographical regions can be used to improve availability. Second, replicas hosted in different locations can be leveraged to provide reduced response times for read queries issued by local clients. Lastly, leveraging locality may also reduce the load on the inter-cloud network links, as merely replication traffic remains to be transferred between cloud data centers instead of the entire client-generated traffic.

2) Large volume data transfer: The High Energy Physics (HEP) scenario described here focuses on the computing requirements of the Large Hadron Collider (LHC) experiments at CERN. The LHC computing resources are distributed to hundreds of independently managed data centers.

Transparent live migrations of VMs running physics jobs between data centers would be one way to keep infrastructure use fair and balanced. The long running LHC computing jobs need continuous network access to the physics data both during and after live migration. From the HEP application point of view the networking environment will not change during the migration process. The network traffic is forwarded to the new location until the job is finished. The high data transfer rates, due to both live migration and the computing jobs, require that network expansion is implemented efficiently without any significant overhead.

[1]https://github.com/hyrise

978-1-5386-3873-6/17/$31.00 © 2017 IEEE

3) Virtual Home Gateway: The Home GateWay (HGW) boxes deployed in telco operators' networks are typically complex devices with advanced network functionality on-board. Due to the evolution of services of operators, several HGW types supporting different sets of functions exist in operator networks. The virtual HGW (vHGW) concept assumes that most of the advanced HGW logic is shifted from the physical device to network resources managed by the telco operator. HGWs could become much simpler devices with only a limited set of functions supported (mainly L2 forwarding), while advanced functions run on VMs instantiated in telco-grade Next-Generation Point-of-Presence (NGPoP) nodes for example.

The federated clouds scenario opens the door for vHGW-related workload migration. The VMs representing customer-serving VNFs (Virtual Network Functions) can be migrated from one data center to another. To that end, appropriate subnets from one cloud can be expanded to others The VNF migration can, for example, be used for network workload re-optimization based on different criteria including load-balancing, energy savings, user activity and traffic volume.

For all these use-cases we assume that the companies involved have their own cloud deployed, independently from the others. However, they are willing to share resources on an individual basis. The clouds, the users and the projects are independent. In case of load bursts or some location specific requirements, the companies will migrate some of their workload to other federated clouds. The network expansion should be transparent to the user (i.e. at layer 2) in order to allow transparent migration and data access.

III. InterCloud expansion agent design

In order to interconnect clouds and support the requirements stated above, an interconnection agent has been developed. This interconnection agent adds multi-cloud awareness and expansion capabilities to OpenStack. In its initial design, the agent proxies the requests for all the OpenStack APIs, redirecting the requests to the correct cloud they should be applied on. All clouds are independent and can be set up differently, even running different versions of OpenStack.

The agents extend the virtual networks provided by Neutron beyond a single cloud location. Agents in each cloud can be connected to several other clouds, without needing a full mesh of interconnections, allowing a model of peering between cloud providers. The agents provide a layer 2 expansion mechanism to allow transparent migration of workloads between the different clouds. The machines can keep the same IP addresses and still belong to the same broadcast domain. The interconnection agent provides the connection and the mechanisms to process the traffic to transparently connect the different parts of the virtual network.

Using L2 expansion adds issues in terms of maximum transmission units (MTUs), which are different for the tunnels connecting clouds compared to the local virtual network. This requires the interconnection agent to perform actions such as clamping the maximum segment size (MSS) for TCP and fragmenting other packets. Fragmentation happens to the encapsulated packets, allowing us to transport oversized non-IP packets.

A. Application interactions

Each agent offers APIs for communication between clouds and with the user. Two types of actions can be performed. The first type is for setting up the agent and is used by the cloud administrator. The administrator will define IKE policies, IPsec policies, and will create the IPsec tunnel and the interconnection through this API. The second type of API is for the user to expand a network into another cloud. All users can expand their networks to the available clouds. The available clouds are defined by the administrator. In future work, it could be based on policies, or criteria set by the administrators.

B. Inter-agent interactions

An IPsec tunnel is established between the agents during the setup by the administrators. Once enabled, the agents from the two clouds are able to communicate using their APIs for the configuration of expanded networks. When a user requests a network expansion from cloud A to cloud B using the API of the agent in cloud A, the agent will forward it to its peer in cloud B to create a virtual network for this tenant and set it up accordingly. The agents use a specific tunnel ID to isolate the traffic of this network. The approach chosen here is to share the endpoints for all the users. Isolation is performed using the tunnel ID. Multicast traffic from a virtual network is forwarded only to those clouds to which the network was extended. The advantage of sharing a single tunnel is the ease of use, since the users do not have to configure their own tunnels. However, the traffic needs to be shaped to avoid situations where a user would use all the available resources at the expenses of all other users.

From the cloud administration point of view, federation of different clouds could be performed by using a project model, where the peer cloud has a project and expands all its networks into this single project, or by replicating the expanded project in the peer cloud. The first approach provides good isolation and ease of metrics collection, but does not provide quotas per project. We are currently using a second approach, that replicates the projects into the peer clouds and that provides good project based isolation and quota enforcement per project per cloud.

C. Centralized or decentralized networks

When expanding a network, two types of approaches are possible, considering the network architecture: centralized or decentralized. In the case of a centralized approach, considering a network from cloud A that would be extended to cloud B and cloud C, clouds B and C could be unaware of each other. Thus, if a VM on cloud B wants to communicate with a VM in cloud C, the traffic could flow through cloud A. This allows the different cloud providers to have federation agreements independently of each other and fully use the resources available. In the decentralized approach, cloud B and cloud C would be aware of each other and be interconnected in a full-mesh, hence permitting the traffic to flow between the two clouds directly without going through cloud A. The advantage of this technique is the reduced latency, as the traffic uses a direct path between the two clouds rather than going through an intermediary. It might also reduce the networking costs depending on the agreement the cloud providers have for their

uplinks. However, a hybrid approach is possible, but has not yet been evaluated.

D. OpenStack network architecture

The network in OpenStack deployments is configured by the networking manager *Neutron* and the computing resource manager *Nova*. Both Neutron and Nova can use different back-ends for enacting their configurations. For example, Neutron can be configured to work with Linux bridges as well as with Open vSwitch.

When using Open vSwitch as the switching back-end, Neutron sets up a specific architecture, composed of several switches. The architecture of Neutron is displayed in Figure 1. The core switch is *br-int* to which all the virtual machines on the compute host are connected. Routers and DHCP servers are also connected to *br-int* [1].

OpenStack allows separating cross-server traffic from tenants in multiple ways. One way is to separate the tenants using VLAN IDs. Another way is to use tunnels, e.g. GRE or VXLAN. Independent of the method used, Neutron uses an additional switch. If using GRE or VXLAN tunnels, a full mesh network of tunnels between all the hosts is created. Those tunnels are connected to a switch called *br-tun*. This switch performs translation between the tunnel IDs and the internal VLAN IDs used in the node, which can be different for each hypervisor or network node. If using VLANs for segmentation, the trunk interface is directly connected to a switch that is performing translation between the segmentation VLAN IDs and the internal VLANs used by Neutron. This switch is connected to *br-int*, too.

E. Cloud interconnection

In order to connect several clouds at layer 2, we have copied and adapted the structure of Neutron. We have added a switch called *br-cloud* that is connected through a patch port to *br-int*. Our agent uses the same VLAN IDs as the Open vSwitch agent to isolate the traffic of the tenants. The tunnels to the other clouds are also connected to *br-cloud*. VMs on different clouds are able to communicate using layer 2 protocols. The architecture of the solution is displayed in Figure 1.

In order to optimize the performance, different tunneling mechanisms can be used, including VXLAN, Geneve and GRE. By default, VXLAN tunneling is used. Moreover, VXLAN or Geneve dynamically assign the source ports. This is particularly useful in case of load-balancing based on UDP and IP headers.

We support the encryption of the traffic between clouds using IPsec encryption. By default, the Encapsulating Security Payload (ESP) protocol is used in transport mode since the tunneling already adds a new IP header. However, if at least one of the clouds is behind a NAT, the NAT-traversal (NAT-T) feature of IPsec is used with ESP in tunnel mode. This is, for example, a specific requirement for our test setup with CERN site. However, using NAT-T increases the overhead and discards the advantages given above, since the UDP header is encrypted and placed in the ESP payload. It is not possible to access the UDP headers information that could be used for load balancing for example in link aggregations protocols. The overhead for VXLAN is 54 Bytes with the VLAN tag, for GRE it is 46 Bytes. IPsec encapsulation creates an overhead of up to 37 Bytes (65 with NAT-T). The maximum MTU that can be used without fragmentation, using VXLAN, on the VM is 1416 Bytes. This is a 2,3% MTU decrease compared to the MTU set by OpenStack with tunneling segmentation (1450B). Considering NAT-T, the maximum usable MTU for the VM without fragmentation is 1384, a 4,6% decrease.

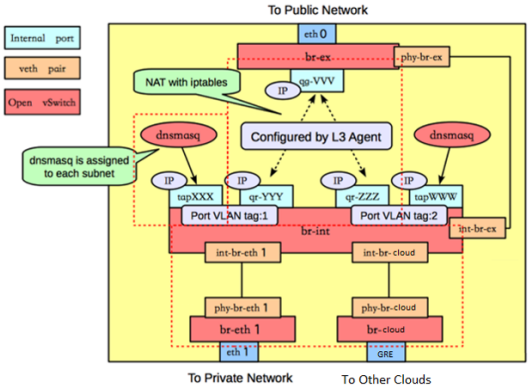

Figure 1. Architecture of Neutron using Open vSwitch and VLANs with the interconnection adapter (based on [2])

In case a cloud needs to be extended to a location without an agreement for expansion in place or to a public cloud, the cloud user might want to expand his cloud network by himself. In order to accommodate this scenario, the architecture of the interconnection agent setup has been slightly modified to run the agent as a standalone service, separated from the OpenStack infrastructure. Thus, the interconnection agent can run in a virtual machine, in any cloud, as long as the virtual machine is directly connected to the networks to be expanded. The services available for cloud providers are also available for the user in standalone mode. However, the interactions between the standalone agent and a legacy agent are limited to prevent the user accessing features without authorization.

IV. Performances measurements

Five partners have set up OpenStack testbeds: Aalto University, Helsinki Institute of Physics (HIP) at European Organization for Nuclear Research (CERN), Hasso Plattner Institute (HPI), Orange Poland and NEC. All of the partners operate clouds for interconnection with some of the other partners. Since the interconnection is usually done over the Internet, throughput is highly volatile. We have thus conducted performance measurements with two clouds in a single site connected through a direct link. The compute nodes are running on bl645c blades, and virtual machines are run using KVM. The controller nodes (including the networking elements, Neutron) run in virtual machines and support AES-NI. The compute nodes have 12 cores and 32GB of RAM, the controllers have 8 cores and 24GB of RAM. The throughput between the controller nodes of the clouds is 10Gb/s, 2.5Gb/s with VXLAN tunneling. The throughput between the compute nodes and the controllers is 1Gb/s. We evaluate several interconnection mechanisms for the sake of comparison:

- VPN as a Service (VPNaaS; OpenStack's cloud expansion mechanism, see Section V for more details), in tunnel mode (ESP);

- VPN as a Service, in transport mode (ESP);
- The interconnection agent, allowing L2 expansion.

For reference, floating public IP addresses provide a throughput of 815,25Mb/s, without encryption nor tunneling. However, a double one-to-one NAT is performed by Neutron. That is the maximum throughput achieved, due to the 1Gb/s limitation between the compute and controller nodes. Similarly, for VPNaaS in transport mode, floating IP addresses are used for the VMs. This means that one-to-one NAT is performed on both clouds. This is not the case for transport mode. The interconnection agent uses VXLAN and IPsec in transport mode (ESP). We also compare those results with an encrypted connection between the controllers (transport mode) and with a tunneled and encrypted connection between the controllers (VXLAN and transport mode). The available bandwidth using TCP is displayed in Figure 2. The measurements were performed using iperf with 10 concurrent connections.

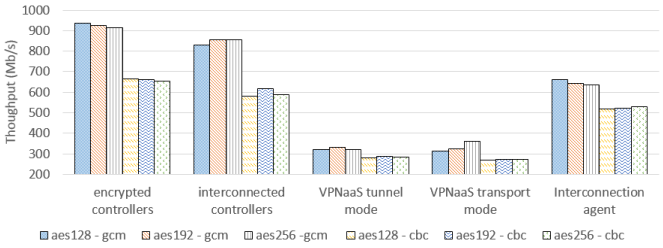

Figure 2. Throughput using TCP depending on the encryption and mechanisms

The interconnection agent performs much better than the VPNaaS for a one main reason. The IPsec software used, Strongswan, is able to make use of multiple cores of the controller node when running for the interconnection agent. However, when running for VPNaaS, with the default configuration of the current versions of OpenStack (Newton), of the VPNaaS agent (9.0.0) and of Strongswan (5.3.5), only a single core is used for encryption in a single namespace. Thus, the performance of the VPNaaS solution when handling several connections is worse than the performance of the interconnection agent. Using VXLAN or Geneve for the interconnection enables Strongswan to be aware of the different connections and balance the load in the most efficient way possible. The best performances are achieved using the Galois/Counter Mode, due to the hardware support.

The round-trip time (RTT), when measured from the virtual machine, is around 3 ms in all the cases, while it is around 1 ms from controller to controller.

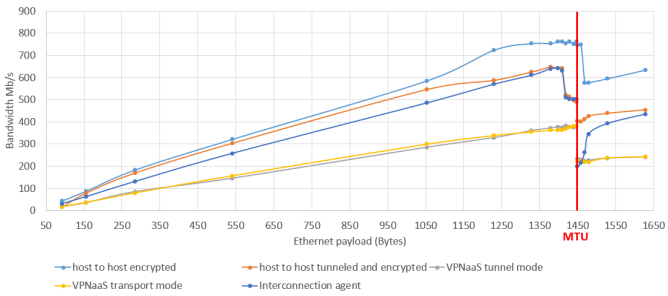

Figure 3. Throughput using UDP depending on the packet size and mechanism

Considering UDP, the possible fragmentation has to be taken into account. As can be seen in Figure 3, the performance of the interconnection agent, using AES256-GCM128 encryption, is still higher than that of VPNaaS, close to the maximum we can get with encryption, but the impact of fragmentation is different. In both cases, the encryption software will fragment packets when their size increases. Due to the VXLAN overhead in the interconnection agent, this fragmentation already happens with smaller packet sizes than for the VPNaaS solution. However, if the size keeps increasing, the VMs will start fragmenting, too. However, when the VM fragments its packets, the first packets still need to be fragmented by the interconnection agent, while with VPNaaS, the packets are reassembled and correctly re-fragmented by the Linux kernel. This double fragmentation is an issue in terms of performance, but the interconnection agent still performs better than the VPNaaS solutions.

V. State of the art

Cloud federation is a trending topic and has received a lot of attention. For instance, CloudNet provides support for live WAN inter-cloud VM migration [3]. However, they rely on a commercial VPN to achieve the layer 2 connectivity. Hence, we assume that we can achieve live WAN VM migration similarly, and focus on developing an extension of the networking framework for cloud federation and its integration with OpenStack. A project specifically related to our work is BEACON. This project aims at solving the issue of cloud federation in general. They consider different use-cases, such as cloud federation, hybrid clouds and brokers. They have designed a federation agent and a federation dataplane that support tunneling between clouds, using different tunneling technologies and encryption. They offer layer 2 and layer 3 expansion of federated network segments. Their agent works with OpenStack and OpenNebula. When integrated into Openstack, their agent requires the Opendaylight SDN controller [4]. The project also uses the OneFlow service [5]. Our work differs from BEACON in several aspects. We focus on the networking interconnection only, providing a much simpler alternative. The basis of the interconnection is the same, using encrypted tunneling. However, our approach differs in the fact that we do not require the clouds to be interconnected in a mesh model when expanding a network. This allows us to use a model of peer-to-peer cloud interconnections, without the need for a global agreement between the parties, fitting the broker approach and the requirements from our use-cases. We also do not require several plugins, modules and additions to OpenStack either, such as OpenDayLight for BEACON, achieving an interconnection with the simplest deployment of OpenStack.

A second project that relates to our work is Tricircle [6]. This project is developed by the OpenStack community. Even if the aim is similar, i.e., federating clouds, the approach taken is quite different. This project cascades OpenStack clouds, thus having a centralized cloud management and ”slave” OpenStack clouds (pods) [7]. Thus, there is a central cloud, with a central Nova, a central Cinder and a central Neutron service. Some Nova and Cinder extensions were developed (now called Trio2o). A Neutron plugin was implemented to provide the interconnection and help with the federation. However, this approach does not fit our requirements since it implies having a centralized management platform while we want to achieve a decentralized service, with different endpoints for

each cloud. Tricircle is mainly aimed at distributed clouds under a single administrative authority. Accordingly, it requires a unique Keystone service or a federated Keystone which is not feasible when working with independent clouds.

Another alternative is the Contrail project [8]. This project is developed by Juniper. An open-source version is available, OpenContrail. However, OpenContrail provides a full replacement solution for the underlay mechanism in Neutron. Thus the solution for cloud interconnection designed for OpenContrail cannot be applied in an environment where OpenContrail is not used, such as our testbeds.

Another project relevant in this context is the VPN-as-a-Service project of OpenStack (VPNaaS) [9]. This project enables the users to create VPN endpoints for their virtual networks and connect to remote VPN endpoints. However, this requires each user to have a publicly reachable IP address (NAT is supported) for each of their endpoints. Our solution allows users to extend networks to another cloud securely, without the need for a public IP address. Since the encryption is configured by the cloud provider's administrator in our solution, the level of security provided can be relied upon while users might configure their endpoints with weak parameters. Our solution also provides layer 2 connectivity for the virtual networks while VPNaaS implements only layer 3 connectivity.

Considering those different projects, our approach differs in several aspects. We use a broker model with peer-to-peer agreements. This means that the clouds do not have to be fully meshed and interconnected, but our approach takes advantage of any agreement between cloud providers. Our setup fits a scenario with completely independent clouds willing to share resources in case of demand burst. Moreover, the tunneling and encryption methods are set up by cloud providers, giving them full control of the setup and removing the constraints for users such as public IP addresses and security requirements. Finally, if necessary, our approach does not require control of the cloud architecture and the interconnection agent could even be run in a standalone mode in public clouds.

Some other projects aiming at integrating and improving the NFV orchestration from a networking point of view are somehow related, such as the OPNFV NetReady [10] project or the X-OS based on ONOS [11]. None of those projects focuses on cloud federation. Nevertheless, the network expansion could be considered as a use-case for these projects, similarly to the L3 VPN use-case for the NetReady project. However, the federation would then be considered as a feature run by the user instead of the cloud provider, with some of the drawbacks previously mentioned.

VI. Future work

Several directions will be investigated in the future. We will first focus on multiple uplinks. We aim at implementing a load-balancing mechanism to split the load across different uplinks, depending on the achievable throughput. Some special emphasis will be put on the balancing mechanism to tackle issues related to multiple paths. Some network diagnosis tools will also be evaluated for integration to provide adaptability based on the multiple uplinks and to be reactive to failures and delays.

We will also focus on developing services for the interconnection, such as traffic shaping. Some integration with existing management systems for cloud federations will be considered to be implemented in the broker. Finally, we will also investigate how to support migration from the networking point of view.

VII. Conclusion

We have developed a cloud interconnection agent that permits federating clouds following the needs of a number of applications which we deem representative of current cloud workloads. The agent uses peer-to-peer agreements between cloud providers in order to provide flexible distributed cloud topologies. Our agent provides expansion on layer 2 which other comparable solutions do not. It provides security for the users and outperforms other encryption solutions by leveraging the use of multiple cores to encrypt the traffic. This agent is a basic building block that enables us to develop more services and features for cloud providers and users.

Acknowledgement & Disclaimer

This paper has received funding from the European Union's Horizon 2020 research and innovation programme 2014-2018 under grant agreement No. 644866. This paper reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information it contains.

References

[1] OpenStack Foundation. (2017, Jan.) Open vSwitch L2 Agent. [Online]. Available: http://docs.openstack.org/developer/neutron/devref/openvswitch_agent.html

[2] ——. (2017, Jan.) Layer 3 Networking in Neutron - via Layer 3 agent and OpenVSwitch. [Online]. Available: http://docs.openstack.org/developer/neutron/devref/layer3.html

[3] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. van der Merwe, "Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines," in *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '11. New York, NY, USA: ACM, 2011, pp. 121–132. [Online]. Available: http://doi.acm.org/10.1145/1952682.1952699

[4] A. Levin and P. Massonet, "Enabling federated cloud networking," in *Proceedings of the 8th ACM International Systems and Storage Conference*, ser. SYSTOR '15. New York, NY, USA: ACM, 2015, pp. 23:1–23:1. [Online]. Available: http://doi.acm.org/10.1145/2757667.2778189

[5] BEACON consortium. (2017, Jan.) BEACON deliverable 3.1. [Online]. Available: http://www.beacon-project.eu/s/D31-Scientific-Report-WP3-a.pdf

[6] OpenStack Foundation. (2017, Jan.) Tricircle. [Online]. Available: https://wiki.openstack.org/wiki/Tricircle

[7] ——. (2017, Jan.) Tricircle architecture. [Online]. Available: https://wiki.openstack.org/wiki/File:Tricircle_architecture.png

[8] OpenContrail. (2017, Jan.) OpenContrail webpage. [Online]. Available: http://www.opencontrail.org/

[9] OpenStack Foundation. (2017, Jan.) Virtual Private Network as a Service. [Online]. Available: https://wiki.openstack.org/wiki/Neutron/VPNaaS

[10] OPNFV Community. (2017, Jan.) OPNFV NetReady wiki. [Online]. Available: https://wiki.opnfv.org/display/netready/NetReady

[11] ONOS Community. (2017, Jan.) XOS: Service orchestration for CORD. [Online]. Available: http://onosproject.org/wp-content/uploads/2015/06/Technical-Whitepaper-XOS.pdf