Management of Forwarding Elements in Network Virtualization using OpenFlow based Controller Intelligence

ABSTRACT

Server Virtualization technology has revolutionized the way computing is performed and services are being offered. The paradigm shift in service offering has triggered lot of requirements on traditional system architecture. Large data centers are being setup in different corners of the world to cater to customer demands at anytime, anywhere. Cloud computing stresses networks for high-density multi-tenancy, virtual machine mobility etc. The isolation requirement of VMs in data centers cannot be fulfilled by VLAN as the Internet is a L3 network and VLAN has got a theoretical limit of 4096 tags.

Network virtualization is the solution for the aforementioned challenges and we propose a way to manage datapath using control plane information gathered using custom developed network services for virtual environment. We used OpenFlow as communication protocol between control plane and data plane.

1. INTRODUCTION

Network virtualization allows creation of multiple isolated network partitions overlaid on top of a common physical network infrastructure. This model helps to create multiple networks, without the additional investment in hardware[5]. It opens up new possibilities by enabling the deployment of different architectures and protocols over a shared physical infrastructure. This allows the coexistence of multiple networking approaches and provides a separation of services and infrastructure [4].

The packet processing framework defines planes and each plane has a separate role and separate responsibility. Data plane also called as forwarding plane is responsible for what to do with packets arriving based on labels and IP header. Control plane is responsible for exchange of routing information and labels which is collected with routing protocols. It consists of complex mechanism to exchange routing information[2].

In traditional networking device the control processes and forwarding functionality reside on the networking device. OpenFlow, a open communication protocol essentially separates these two functions the fast packet forwarding (data path) and the high level routing decisions (control path) in an OpenFlow switch. The data path portion still resides on the switch, while high-level routing decisions are moved to a separate controller, typically a standard server. The OpenFlow Switch and Controller communicate via the OpenFlow protocol[3].

The rest of the paper is organized as follows. We have described about limitation in current approaches and newer proposals in section 2. In section 3 a brief introduction to our system model is discussed. Implementation details are documented in section 4.

2. LIMITATIONS OF EXISTING VIRTUAL NETWORK MODELS

The implementation of almost all other network services (e.g., policy routes, ACLs, QoS, isolation domains) relies on topology-dependent configuration state. Management of this configuration state is cumbersome and error prone – adding or replacing equipment, changing the topology, moving physical locations, or handling hardware failures often requires significant manual reconfiguration.

2.1 Limitations of VLAN and VRF

Virtualization is not a new thing to the networking domain as networking supports methods like VLANs(Virtual Local Area Networks), VRF(Virtual Routing and Forwarding). However, these network virtualization technologies have not significantly changed the operational model of networking, and administrators of networks continue to write the scripts in order to achieve a limited degree of automation. Present network virtualization techniques have some limitations. These include - VLANs are limited to a broadcast domain, MAC based VLAN requires the managerial overhead to manage the network[19].

2.2 VXLAN, NVGRE and STT

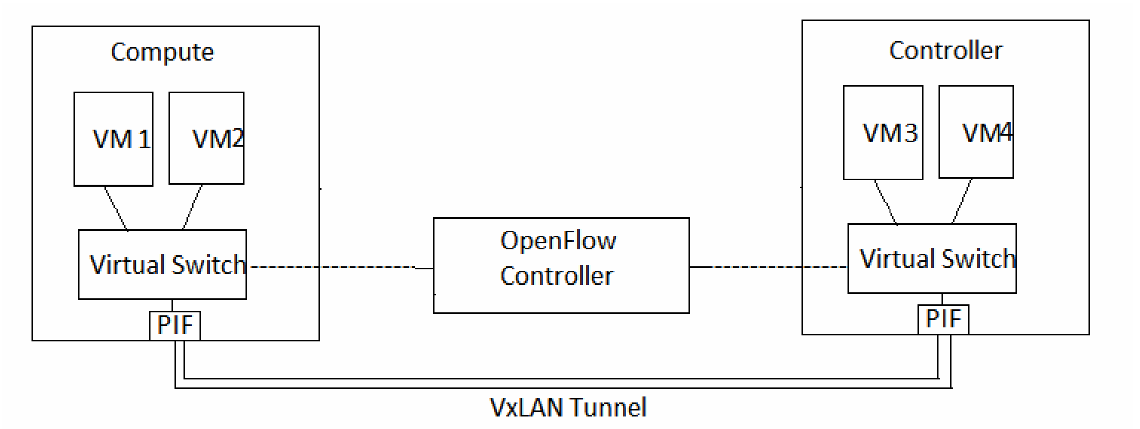
VXLAN is a Tunnel protocol that encapsulates layer 2 connections (Ethernet) in a Layer 3 tunnel (IP). The tunneling of VXLAN allows Local virtual networks (L2) to be extended across different IP sub-networks and enable the migration of virtual machines across different IP subnets. The use of segment identifier of 24-bit length makes the network isolation and segmentation more scalable than the VLAN which uses labels of only 12-bit long supporting only 4096 different VLANs. NVGRE is very similar to VXLAN[17]. The idea of NVGRE is to use the lower 24 bits of GRE key field to indicate the Tenant Network Identifier[16]. NVGRE also uses physical multicast for logical broadcast like in VXLAN. The difference between the two standards VXLAN and NVGRE exists in the way of locating the destination address and in load balancing. VXLAN uses random port number assignment which is used to spread the load while NVGRE uses key field reversed bits. STT is an encapsulation format for network virtualization which is used for end-point-to-end-point tunneling by using TCP Segmentation Offload Capabilities. Unlike the above mentioned protocols, it was designed to be used with soft switching within the server which uses vswitches in their hypervisors by taking advantage of hardware acceleration at the NIC. The main goal of STT is to preserve the flexibility and development speed of software while still providing hardware forwarding speeds[18].

The goal of all of above mentioned protocols is to virtualize the physical network topology and bring the functionality like isolation of multiple tenants, isolation of overlapping address space between multiple tenants and enhanced VM mobility by providing L2 services over an L3 network. We used VXLAN in our experiments[14].

2.3 Quantum

OpenStack is an Infrastructure as a Service (IaaS) cloud computing project by Rackspace Cloud and NASA and now being supported by many companies across the industry. Quantum, part of the NetStack suite, provides "Network Connectivity as a service" to instances running in Openstack clouds. It exposes a generic and extensible API allowing users to build and manage their networks, and uses a pluggable architecture, thus enabling different technologies to implement the logical abstractions exposed by the API. It can be therefore regarded as the fundamental building block for cloud networking services for Openstack, as consumer will be able to create the network topologies required for deploying their applications in the cloud, and build higher-layer network services, such as VPN access, Firewall, or Load Balancing, on top of the virtual networks defined by Quantum[12].

3. SYSTEM MODEL



We setup a multi node OpenStack system with one controller node and one compute node[13]. Each node is having two VMs which are connected to an open vSwitch[11]. Open vSwitches are remotely controlled by OpenFlow controller[10]. There are two networks or subnets and both span across hosts i.e. compute node and controller node. In the figure above, VM1 on compute node and VM3 on controller node form one network and similarly VM2 of compute node and VM4 of controller. We used KVM-QEMU based hardware virtualization[20].

The vSwitches on both nodes forward packets to OpenFlow controller if suitable flow entries are not available in corresponding vSwitches. Custom developed network services run on top of OpenFlow controller. These network services analyze the packet and make intelligent decisions regarding packet delivery, which is framed as a flow entry and used to program the vSwitches.

4. IMPLEMENTATION

We developed networking services such as DHCP Relay and ACL on top of Floodlight OpenFlow controller. The idea is to make use of this intelligence to program switches. Using Floodlight controller we pushed flows and using VXLAN configured tunnels so that VMs can communicate. The main branch of open vSwitch is having support for only GRE tunneling but we experimented with VXLAN tunnels (we used VXLAN specific branch of Open vSwitch in GIT which has not merged with main branch)[15]. We made changes in OVS agent code to handle VXLAN tunnel types. The default tunnel type, GRE was hard coded in existing codebase we enhanced by adding 'tunnel type' parameter in the configuration file and adapted in related methods.

DHCP Relay is actually running on top of the controller. A listener has been designed to get only DHCP packets from VMs and parse the DHCP packets to get the VM details (MAC address). This information is persisted in database. Then the listener will send the DHCP discover packet to the relay agent. The relay agent will prepare a new DHCP packet by adding link selection options and GIADDR field. This packet is sent to the DHCP server. The server will reply with the DHCP offer packet to the relay agent. This packet is sent to the controller by the relay agent. Then the controller will push this packet to the OVS and send out from the port to which the DHCP request packet came. In this way the VMs can get the ip address using openflow controller.

In ACL we have implemented an application such that it will read a configuration file and perform deny or allow of packets between certain set of VMs. As per the policies mentioned in the configuration file, flows are created and pushed by the controller. Tunnels are established or restricted between machines based on policies.

5. CONCLUSION

As traditional networks do not provide network-wide control abstractions, it imposes lot of restriction to innovation and scalability. We conducted studies to explore the possibilities in this direction and were able to successfully use control plane information to define datapath in virtualized networks. At the time of writing this, Essex version of OpenStack got released and it has been used for our experiments. We used Quantum Manager to create virtual networks. The

OpenStack Nova and Quantum interaction has been studied in depth right from launching of VM to termination.

6. ISSUES IN PROPOSED SYSTEM

OpenStack Quantum was an incubating project in Essex with minimal functionalities. There are lot of enhancements planned in Folsom release and which will support network virtualization in more sophisticated way. We have to fine tune our proposal to leverage the capabilities Quantum offerings. The VXLAN overlay tunneling is just a proposal submitted to IETF and waiting for proposal. The proposal is still not mature as the control plane is missing in VXLAN. There is no out-of-band mechanism that a VXLAN host could use to discover other hosts participating in the same VXLAN segment, or MAC addresses of VMs attached to a VXLAN segment. As per the VXLAN draft, the VXLAN identifier to IP Multicast mapping remains a management plane decision and so VXLAN depends on IP multicast to discover MAC-to-VTEP (VXLAN Tunnel End Point) mappings and thus cannot work without an IP-multicast-enabled core[17].

7. ACKNOWLEDGEMENTS

We would like to thank our guide Prof. Shatrunjay Rawat for his constant encouragement and discussions. We thank Aravinda Potluri, MD OneConvergence Pvt. Ltd. for mentoring us and giving us clarity on different aspects of our project including problem definition. Finally we would like to extend our sincere gratitude to our friends, family and IIIT family.

8. REFERENCES

[1] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker. Extending Networking into the Virtualization Layer. In HotNets, October 2009.
[2] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker. Virtualizing the Network Forwarding Plane. In Proc. Presto, November 2010.
[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM CCR, 38(2), 2008.
[4] J. Carapinha and J. Jimenez, "Network virtualization: a view from the bottom," in Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures. Barcelona, Spain: ACM, 2009, pp. 73–80.
[5] Managing Network Virtualization with Virtual Network Manager, http://www.cisco.com/en/US/prod/collateral/netmgtsw/ps6504/ps6528/ps2425/white_paper_c11-541238-00.pdf

[6] Open Networking Foundation, https://www.opennetworking.org/
[7] Open Flow, http://www.openflow.org/
[8] http://networkheresy.wordpress.com/
[9] Nicira NVP, http://nicira.com/en/network-virtualization-platform
[10] OpenFlow Floodlight Controller, http://floodlight.openflowhub.org/
[11] Open vSwitch, http://openvswitch.org/
[12] OpenStack Quantum, http://blogs.citrix.com/2011/06/22/quantum/
[13] OpenStack, http://devstack.org/guides/multinode-lab.html
[14] http://blog.ioshints.info/
[15] https://github.com/mestery/ovs-vxlan.git
[16] NVGRE RFC draft, http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-00
[17] VXLAN RFC draft, http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-00
[18] STT RFC draft, http://tools.ietf.org/html/draft-davie-stt-01
[19] http://tools.ietf.org/html/draft-narten-nvo3-overlay-problem-statement-01
[20] KVM, www.linux-kvm.org/