Managing multi-tenant services for software defined cloud data center networks

Casimer DeCusatis, Robert Cannistra
Dept. of Computer Science and Mathematics
Marist College
Poughkeepsie, NY USA
casimer.decusatis@marist.edu,
robert.cannistra@marist.edu

Ludovic Hazard
Communications Sector Research
IBM Corporation
Frankfurt, Germany
ludovic.hazard@fr.ibm.com

***Abstract* — As cloud computing data centers grow larger and networking devices proliferate, many complex issues arise in the network management architecture. We propose a framework for service offerings in a dynamic cloud network, based on software defined networking principles. The proposed approach also facilitates the use of multi-tenancy applications by providing redundant, vendor agnostic zone controllers with a hierarchical network manager. This approach enhances network security and supports converged Fibre Channel over Ethernet storage. Experimental implementation is demonstrated in a multi-tenant cloud test bed using a hybrid environment including both KVM and VMWare components.**

Keywords— cloud; software defined network; network function virtualization; Fibre Channel over Ethernet; hypervisor

I. INTRODUCTION

Traditional information technology (IT) infrastructures manage servers, storage, and networking as homogeneous silos, with extensive manual intervention by highly skilled system administrators. Virtual resources are also statically configured and often managed in the same manner as the hardware resources which they replace. This hardware-centric management approach is no longer sustainable, due to the emergence of dynamic, multi-tenant cloud services, growing applications such as mobile computing or big data analytics, and the need to constrain physical resources (avoiding over-provisioning) to contain growing capital and operating expenses. A new approach to workload aware management is emerging in which servers, storage, and networking are treated as pools of fungible resources which can be assembled and managed as a single entity, using software virtualization and advanced automation tools. This leads to a more flexible, elastic service delivery model which radically improves time to value and availability for new service deployments. Higher utilization provides efficiency gains which lower the total operating cost. Automation improves the repeatability, consistency, and stability of the environment, and enables workload optimization.

This environment is characterized by a programmable infrastructure using open application programming interfaces (APIs). Services are dynamically assigned to the best set of available resources based on characteristics of the application. Programmable APIs enable continuous optimization to quickly address infrastructure issues (such as providing self-healing or workload scaling services) and make the environment more responsive to business needs. A key element of this approach is multi-tenant services, which leverage virtualization and abstraction technologies to provide different users with secure access to applications which reside in a shared environment. Multi-tenancy has become an important part of the $130 B cloud computing market [1].

New cloud service offerings and revenue opportunities can be realized by leveraging a flexible, on-demand infrastructure. A significant bottleneck in this area is currently being addressed with the introduction of software defined networking (SDN) and network virtualization tools [2]. SDN network management involves separating the data and control/management planes within a switch, and using a centralized network controller to oversee the entire network. These features enable faster service provisioning and lower costs due to the resulting automation and efficiency gains. The network and control planes can now be developed and optimized independently, and the use of standardized management APIs with vendor-specific plug-ins helps avoid vendor lock-in. Network applications can be more easily deployed on top of a centralized controller, as opposed to traditional approaches which need to change the switch firmware or control logic to implement new services. Virtual networks can also be used to interconnect virtualized network functions (VNFs), which replace capital intensive, manually provisioned services with automated, highly virtualized software applications running on commodity servers [3]. However, the use of SDN and VNF in multi-tenant clouds is a relatively new topic, and there is a need for a network management architecture which addresses the unique service requirements of this environment.

In this paper, we present a management architecture and experimental test bed demonstrating deployments of multi-tenant management environments. Although the management

978-1-4799-4998-4/14/$31.00 © 2014 IEEE

framework is hypervisor agnostic, the test bed illustrates the co-existence of VMware and KVM implementations with multiple network controllers. Although similar functions can be implemented with either hypervisor, we demonstrate that both types can co-exist in an industry standards-based implementation. This is a practical issue for many deployments, which may use different hypervisor environments for different parts of their data centers. Further, this approach enables data centers which may need to merge environments running different hypervisors, without reconfiguring any existing data center hypervisors. SDN is enabled for both physical and virtual flow control for multi-tenant cloud computing.

II. Network Service Requirements

As data centers grow larger and networking devices proliferate, system architects are forced to give closer attention to the complexity associated with network management. There are several technical concerns with SDN that will be addressed in the remainder of this paper. First, SDN requires a centralized network controller, but the current standards provide little information on how to achieve redundant controller instances for high availability. Second, there are a number of different controllers which have been proposed (including Open Daylight, NOX, Floodlight, and several vendor proprietary versions) as well as different server hypervisor environments (KVM, VMWare, etc). These controllers lack a common northbound or application level API, making it difficult for more than one type of controller to exist within the same network. This has forced many users to choose between controllers, each of which has its own strengths and weaknesses, and let to fragmentation of open source controller development efforts. Finally, it is difficult to insure data plane security in an SDN network, since all packets with the same type of flow are handled in the same way. If an attacker determines the flow characteristics, there is no standard mechanism to prevent attacks based on packet injection.

We will also address several other network management issues facing large data centers and cloud computing services, particularly related to multi-tenancy, scalability, and integration of IP and FCoE storage. Large, highly virtualized data centers (including public and private clouds) need to share data center resources across multiple tenants. In a private cloud, for example, different departments in the same organization (research, sales, marketing, etc.) may need to share resources but remain logically isolated from each other for security and performance reasons. Similarly, in a public cloud multiple companies may need to share resources while maintaining separation of their network traffic. This requires a mechanism to isolate network data flows from each other, preferably end-to-end across the entire network (including virtual switches in the server hypervisor). In an SDN network, multi-tenant isolation would be performed by the network controller. The controller may dynamically isolate physical flows using a protocol such as OpenFlow. Alternately, the controller can manage a network overlay (such as IETF standard NV03 or VMWare's ESX/NSX offerings) to create secure tunnels across the network with virtual layer 2/3 addressing. In both cases, network management functions can be automated and saved as a profile or pattern, which may be reused on other parts of the network or customized for other applications. Network patterns are a good way to avoid manual configuration errors, particularly when configuring network security policies. A virtual network overlay has the advantages of being provisioned entirely in software, without requiring any physical re-wiring of the network switches. To provide cost-effective scalability, a network overlay should also map virtual networks into their corresponding virtual servers with some form of directory service, analogous to the DNS management system used on the Internet [4, 5].

Further, network controllers in a multi-tenant environment need to manage Ethernet/IP or Fibre Channel over Ethernet (FCoE) attached storage networks in addition to server networks. The convergence of IP and dedicated storage area networks offers potential benefits in terms of simplified management and cost savings which result from using a common network infrastructure for multiple applications. Since FCoE protocols encapsulate Fibre Channel protocols into an Ethernet frame, routing is still handled by a Fibre Channel Forwarded (FCF) embedded somewhere within the network. It is desirable to control and provision storage connectivity as part of an overall framework for network management.

III. Network Management Architecture

The proposed network management architecture is illustrated in Fig. 1. The approach of dividing the management functions into four layers builds on recent proposals for a single administrative domain with multiple controllers [6]. Each layer is connected with the one above it using a management API; we have illustrated one example showing how OpenStack can be deployed in this environment.

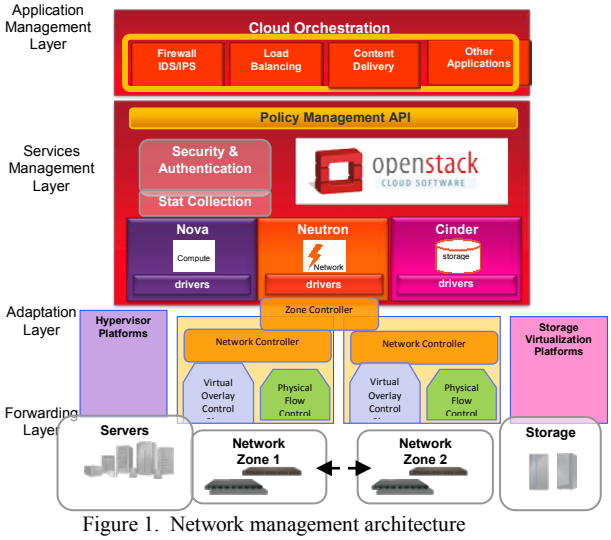

Figure 1. Network management architecture

The architecture shown in Fig. 1 uses four functional layers (forwarding, adaptation, services management, and application

management, as well as multiple controllers (each with a partial view of the network) which was previously shown to yield improved performance and robustness compared with alternative designs [6]. However, this approach differs from prior proposals through the use of OpenStack middleware for the adaptation, services, and management layer, which was not proposed previously. Advantages of this approach include the use of a single policy management API which governs security procedures and statistical data collection from the underlying physical layer. Further, the industry standard OpenStack northbound API is more readily integrated with a higher level cloud orchestration layer. We also extend prior approaches by incorporating multiple hypervisors, a virtual overlay network, and support for FCoE storage. These features add significant versatility to the resulting architecture, and enable new functionality (for example, co-existence of multiple hypervisors during data center mergers or use of network overlays to enhance multi-tenant isolation). We note that previously published architectures [6] provided only a theoretical comparison between different design tradeoffs; in section III, we have created the first experimental implementation of this approach.

At the lowest or forwarding layer, the network is logically partitioned into zones, each of which has its own network controller. Some form of multi-controller solution is desirable to avoid a single point of failure, as well as to avoid latency and performance bottlenecks when the controller is located far from parts of the network. The proposed architecture avoids these issues by distributing control functions across multiple controllers, each of which controls a local segment of the network [7-9]. In an extended distance metropolitan area or wide area network, zones can be chosen based on geographic location of the network segments in order to reduce latency between the network and controller. Dividing the network into zones should also reduce the management traffic load on each controller; this also facilitates performance-efficient scalability. Since the number of devices per zone can be kept fairly small, the controller can poll them more frequently to obtain more accurate topology and configuration information.

The second or adaptation layer of the architecture includes zone management functions which accommodate different types of northbound and southbound APIs in the same network. This also enables the use of different brand controllers for each network segment. For use cases which require an end-to-end overview of the entire network, a higher level management API can provide direction to all the underlying controllers. This vertical hierarchy approach, which has been used by other SDN controller [7], should reduce latency between the network devices and controller in most situations, and simplifies network management rules for the higher level controller. For example, the higher level controller might be used to re-route elephant flows across the entire network, while an elephant flow between devices in the same network zone could be handled by the local zone controller. Since the architecture enables frequent polling from the zone controllers, aggregate configuration data can be forwarded from the zone controllers to the adaptation layer as required to support end-to-end decision making. The proposed approach may use a combination of inband and outband management networks; note that this approach is independent of the method used to communicate between controllers.

Above the adaptation layer, the third or service management layer can be implemented using open source middleware such as OpenStack [10]. The OpenStack network management API, known as Neutron, can interface with the upper level hierarchical controller, which in turn manages the zone controllers. Similar APIs (Nova for servers and Cinder for storage) allow for the orchestration of resources beyond the network. Note that while the Cinder API manages file and block storage devices, it does not include storage area network management or an FCF interface; these functions must be provided through the controllers under the Neutron API. The service management layer provides functions including topology management, security credential management (OpenStack Keystone), interpretation of the management API messages (OpenStack Congress), and aggregate statistical monitoring. In particular, security features can be implemented such as automated authentication when a new device attempts to connect to the network. OpenStack would maintain the public keys for controllers and network devices, insuring that only authenticated controllers can connect to the management system. Security between the controller and network devices is handled by the switch API protocol (OpenFlow, VMWare NSX, or something similar).

Finally, the fourth or an application management layer provides integration of other network management applications, including firewalls, intrusion detection and prevention systems, load balancers, and more. A RESTful management API provides loose coupling between the application and service management layers, and offers a wide range of features for the service management layer. A loosely coupled approach allows for stand-alone network appliances, such as physical and virtual firewalls, to use their native management interface features while also taking advantage of the latest network topology and configuration information propagated from lower layers of the management model. Application management can also be realized using tools which supervise and abstract many underlying network devices, such as IBM Smart Cloud Orchestrator (SCO). For large cloud environments, administrative tools may be used to facilitate installation of many host bare metal operating systems on an entire rack of servers, all at once.

IV. Test Bed for Multi-Tenancy Management

We have successfully implemented a software-defined orchestrator, which includes support for FCoE, overlay virtualization (IETF draft standard NV03, implemented using IBM SDN Virtual Environment), and physical flow control using a Layer 3 OpenFlow network. To our knowledge, this is the first time these features have been successfully deployed

together using the OpenStack reference architecture illustrated in Fig. 1. The network underlay, which is shown in Fig. 2, consists of three adjacent data centers connected by a fully meshed OpenFlow network with software-defined control of Layer 3 functionality. The IP addressing scheme is shown in Fig. 2. Any standardized controller uses OpenFlow 1.3 or higher can be used; our experiments did not reveal a significant performance difference for commonly supported features. The associated FCoE storage network uses IBM/Lenovo G8264-CS 10/40G OpenFlow switches with an embedded FCF manager, interconnecting a cluster of four x86 servers.

This test bed is a hybrid environment, using VMware ESXi as the hypervisor and FCoE connection management, as well as the orchestrator running in a KVM environment. The environment was also tested using different SDN controllers, including Open Daylight (Helium release), FloodLight (Nightly release), and the IBM/NEC branded controller (release 2.3). Additional hardware and network configuration details are provided in Fig. 2 and 3. This implementation requires a shared management network for both hypervisors. In other words, the orchestrator uses direct IP access into each VM to check status, validate the deployment, and apply network configurations (for example, scripting for a firewall, load balancer, or other Layer 4-7 appliance). Consequently, we implement a management network connected to the orchestrator which is also shared by all tenants. To maintain multi-tenant isolation, we create one management network per tenant connected through a firewall to the shared management network, as illustrated in Fig. 3. Further, we configure the overlay with SDN VE (February 2014 release), providing multi-tenant isolation from the hypervisor virtual switch throughout the rest of the network. This allows us to move VMs across Layer 3 boundaries without requiring large Layer 2 domains or special network protocols such as TRILL or SPB [11, 12]. The overlay directory service maps VMs into virtual networks, facilitating cost-effective scaling which can theoretically extend to thousands of VMs per physical server and over tens of thousands of virtual networks.

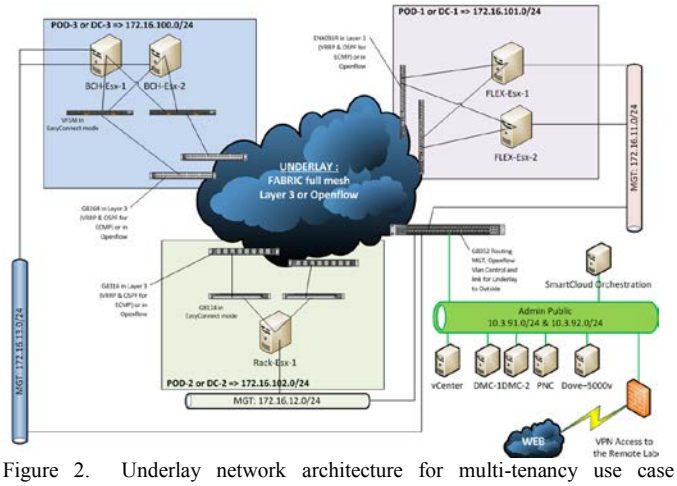

Figure 2. Underlay network architecture for multi-tenancy use case implementation

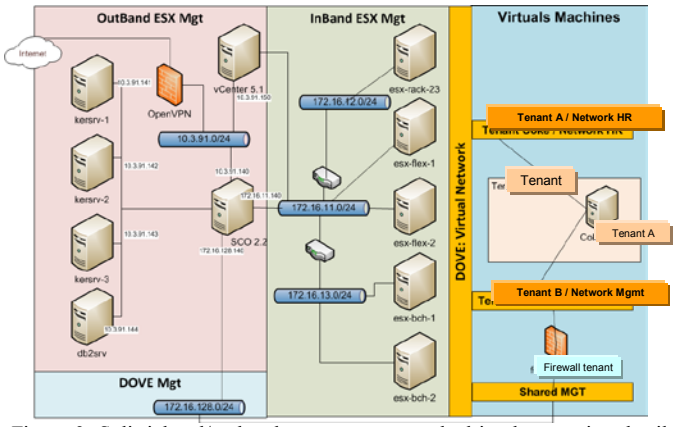

Figure 3. Split inband/outband management test bed implementation details

The orchestrator has been used to develop new network service offerings such as the creation and assignment of new networks and subnets for overlays. The RESTful API is used to create domains, networks, and subnets, and to create associations between subnets, networks, and gateways. Then we create the shared management network, interconnect the virtual firewall, and configure scripts for the virtual firewall (in this case we used Brocade's Vyatta 5400) which provide secure access to the new management domain. Next, VMs are deployed with one virtual network interface card (vNIC) in the management domain; the vNIC configuration is validated, and if necessary the VMs are configured by a separate automated script which we have created for this purpose. In addition, virtual layer 4-7 appliances can be quickly deployed as waypoints to form function graphs or service chains. Initially this was implemented as shown in Fig. 4 using automation based on VMware vCenter 5.0 with an Ethernet overlay and the appliance management interfaces. We have shown that this process can be automated, saved to a service pattern catalog and re-deployed across multiple instances of the network. This improves consistency, reduces or eliminates human error when redeploying the same pattern, and enables rapid, automated reuse of the pattern once it has been created.

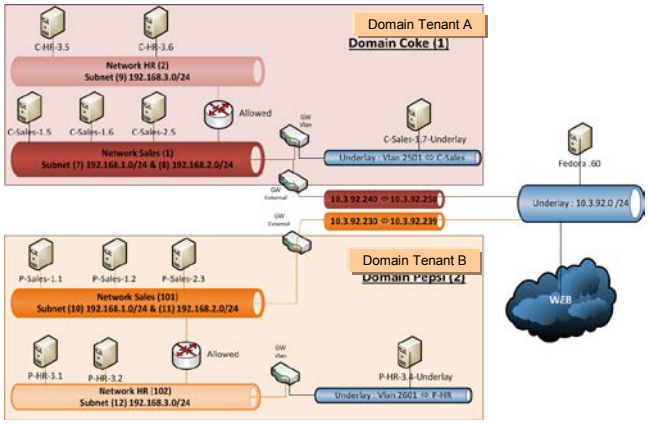

Figure 4. Multi-tenant management use case implementation

V. Conclusion

We have demonstrated a management, provisioning, and orchestration framework for hypervisor agnostic multi-tenancy in a large data center. Our approach is based on the OpenStack Neutron interface for network orchestration and SDN physical and virtual flow control. Using this approach, dynamic resource provisioning has been demonstrated in under a minute (as opposed to conventional approaches which require days or even weeks). There are several proposed extensions to this interface currently under consideration which would improve our management capabilities. For example, industry standard certification programs for OpenStack are still under development (a recent proposal from the Ubuntu OpenStack Interoperability Lab may be a first step towards addressing this concern). Presently, the orchestration API implementations differ by vendor for each OpenStack release; features which work on the Grizzly release, for example, may not work properly with the Havana release, or the same feature from two different vendors may not work the same way even if both are implemented under the Havana release. There are related issues with the Linux distro and kernel levels supported by different vendors, including name space support in the KVM kernel and support for Heat processes at the distro or hypervisor level.

Our work has identified a number of issues related to network management and performance which should be the subject of future research contributions to upcoming OpenStack releases. For example, Neutron plugins such as Open vSwitch 1.4 use only one core in a multi-core processor to match flows in a flow table; since this does not take advantage of multi-core processing potential, it can result in a potential communication bottleneck. For converged IT and CSP infrastructures, OpenStack management faces challenges including resource scheduling across multiple data centers and traffic bifurcation issues (i.e. management of network traffic which is allowed to use available sub-rate channel capacity, rather than filling the entire available channel bandwidth). Future work will also include performance evaluations for different types of workloads, such as determining whether this general purpose architecture implementation can be optimized for use with specific applications such as big data analytics or high performance computing.

References

[1] Gartner Group reports, "Public cloud forecast" (June 2013) and "Private cloud matures" (September 2013), www.gartner.com (last accessed October 9, 2014)

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parklkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks", ACM SIGGCOM Computer Communications Reviw vol 38 issue 2, p. 69-74 (April 2008)

[3] ETSI General Specification NFV 001 - 004 ver 1.1.1 (October 2013) http://www.etsi.org/technologies-clusters/technologies/nfv (last accessed October 9, 204)

[4] L. Dunbar and D. Eastlake, draft IETF NV03 working group standard reference, Sept. 20, 2013 https://ietf.org/meeting/88/agenda/nvo3-drafts.pdf (last accessed October 9, 2014)

[5] VMWare ESXi bare metal hypervisor, http://www.vmware.com/products/vsphere-hypervisor (last accessed October 9, 2014)

[6] R. Ahmed and R. Boutaba, "Design considerations for managing wide area software defined networks", IEEE Communications Magazine p 116-123 (July 2014)

[7] S. H. Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications:, Proc. HotSDN 2012, p 19-24 (2012)

[8] T. Koponen et.al., "Onix: A distributed control platform for large scale production networks:, Proc. OSDI 2010, p. 1-6 (2010)

[9] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distriucted control plane fo rOpenFlow":, Proc. 2010 Internet network management conference Research on Enterprise Networking, ser INM/WREN 2010, Berkeley, CA p. 3-13 (2010)

[10] OpenStack: open source software for building public and private clouds, www.openstack.org (last accessed October 9, 2014)

[11] IETF Working Group, Transparent Interconnection of Lots of Links (TRILL), http://datatracker.ietf.org/wg/trill/charter/ (last accessed October 9, 2014)

[12] IEEE 802.1aq, Shortest Path Bridging (SPB), http://www.ieee802.org/1/pages/802.1aq.html (last accessed October 9, 2014)