DFVisor: Scalable Network Virtualization for QoS Management in Cloud Computing

Lingxia Liao, Victor C.M. Leung, and Panos Nasiopoulos
Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada
liaolx@ece.ubc.ca, vleung@ece.ubc.ca, panos@ece.ubc.ca

***Abstract*—Increasingly cloud-based virtual networking environments are required to provide fine-grained Quality of Service (QoS) management without sacrificing scalability. However, no single approach currently can meet these requirements simultaneously. This paper introduces a layered concept that uses a common overlay mechanism to virtualize networks and enable fine-grained QoS management through resource slicing. Based on this mechanism, a fully distributed network virtualization platform called Distributed FlowVisor (DFVisor) is proposed. It uses a layered overlay to improve the network addressing space, reduce flow setup latency, and remove the single point of failure in the network - the central slice controller. Through a distributed synchronized two-level database, DFVisor incorporates a push-based flow setup and statistics collecting mechanism using a dedicated data channel to address the scalability issues caused by the current pull-based mechanism and the limited control channel bandwidth shared by both control flows and network statistics. The potential issues in DFVisor implementation and evaluation are discussed for future research.**

***Index Terms*—OpenFlow, cloud computing, network virtualization, QoS**

I. Introduction

With the rapid development of cloud computing, more and more applications are migrating to the cloud to take advantage of its elastic resource pool and pay-as-you-go charging model. The large number of applications that belong to different tenants of a cloud data center may have different Quality of Service (QoS) requirements, which creates a big challenge for cloud virtualization environments to provide fine-grained network QoS management while maintaining scalability. Currently, there is no single approach that can achieve these goals simultaneously. The most commonly used approach of cloud network virtualization is overlay, and in current research fine-grained QoS management is often based on flow switching through OpenFlow protocol.

Overlay uses tunneling techniques to enable network virtualization. It nicely addresses scalability in large scaled network environments with complex network topologies, and has been adopted to enable multi-tenant network virtualization environments in current cloud data centers. However, current overlay implementations are mostly based on the existing network protocol stack, which makes it difficult to support fine-grained QoS management. OpenFlow based flow switching is a mechanism that has been studied recently to enable fine-grained QoS management for OpenFlow network fabric. An OpenFlow supporting switch can differentiate flows and hence support fine-grained QoS management. This mechanism (referred as resource slicing in this paper) has been explored to provide virtual networks with different QoS to different tenants [1]. Using this mechanism, each virtual network is defined as a slice such that two slices are logically isolated through the slice controller, and the flows of a slice are forwarded according to the flow entries. Therefore, it facilitates a multi-tenant network virtualization environment with information isolation at network nodes. It nicely addresses the issue of the lack of fine-grained QoS management support in overlay techniques. However, this mechanism depends highly on the OpenFlow protocol and the centralized slice controller model, and has the following limitations: 1) the address space of the 12-bit virtual local area network (VLAN) identity is limited; 2) the current pull-based flow setup procedure supported by Openflow switch specification creates flow setup latency; 3) sharing of limited control channel bandwidth between a switch and its controller by both control and network statistics flows limits the flow setup rate; 4) the statistics gathering latency caused by current pull-based statistic gathering procedure is not able to support some global flow schedulers [2]; 5) the centralized slice controller used in the current resource slicing approach such as FlowVisor adds more flow setup latency; 6) a centralized slice controller represents a single point of failure in the network. Current research shows that these six limitations can cause a big scalability issue in a cloud data center. Finding an approach for a cloud network to provide fine-grained QoS management without sacrificing scalability is significant in both research and practice.

This paper introduces a layered overlay concept that constructs multiple OpenFlow overlays on top of a common overlay. Through combining tunneling and resource slicing virtual network techniques, it ensures that only selected slices and the selected flows in a slice are forwarded using flow based switching, such that problem (1) can be solved by the mature overlay techniques, and problems (5) and (6) can be lessened through devolving the functions of the centralized slice controller to a local Virtual Network (VN) slicer module in each switch and enabling the control traffic to bypass the conventional switch data path. Based on this concept, we present the Distributed FlowVisor (DFVisor), a scalable cloud network virtualization platform for a cloud provider with an Openflow enabled Cloud network to support fine-grained QoS management. In addition to the layered overlay, the proposed DFVisor also introduces a distributed synchronized two-level database system including a distributed global database and

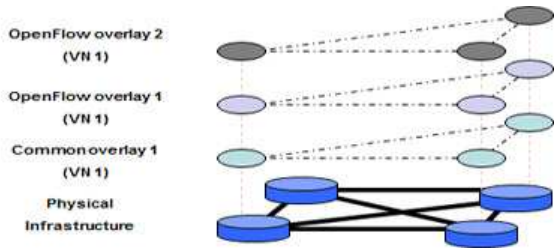

Fig. 1. The Layered Overlay Mechanism.

multiple local databases for each switch and switch controller based on Apache Zookeeper [3], a high-performance coordination service for distributed applications. Through its watch service, the data of global and local databases can be kept synchronized without invoking any other protocol. Our proposed DFVisor platform not only maintains a global network view to simplify the whole network configuration and management but also enables a push-based flow setup and statisticadditive white Gaussian noise at the BS and the k-th eavesdropper, the reces gathering approach to improve the network scalablity that has so far been limited by flow setup and statistics gathering latency. It also creates a dedicated data channel for network configuration and statistics collection such that the whole OpenFlow control channel between a switch and its controller can be used for flow controlling. In this way, our proposed DFVisor eases the network scalability issues caused by problems (2), (3), and (4).

This paper presents a research work in process with three major contributions: 1) we introduce a layered overlay concept to support fine-grained QoS management and scalability at switches; 2) we propose maDFVisor, a network virtualization platform which features a fully distributed architecture based on a distributed synchronized two-level database to support the implementations of the layered overlay and distributed slice controller module in each switch to address the six scalability issues mentioned above; 3) we analysis the potential issues in the implementation and evaluation of DFVisor.

The rest of this paper is organized as follows. We introduce the layered overlay concept in Section II, and then propose F)the DFVisor platform in Section III. We discuss the potential issues in DFVisor implementation and evaluation in Section IV and summarize the related research in Section V. Conclusions are drawn in Section VI.

II. Layered Overlay Mechanism

The current resource slicing approach uses a centralized slice controller, which is situated between the network controllers and the switches to monitor and manage the flows. This added middle box ensures the information isolation among virtual networks.

It has been identified that the current resource slicing network virtualization approach lacks a native network addressing mechanism and causes scalability issue in large-scaled cloud networks with complex network topologies. However, not all the flows in a cloud virtual network need fine-grained QoS support. Simply using resource slicing mechanism to facilitate both virtual network and fine-grained QoS management is inefficient. Managing all the flows of a virtual network with fine-grained QoS is not only unnecessary but also wastes resources. Based on these observations, we introduce a layered overlay concept. By combining the common overlay and resource slicing mechanisms, our approach decouples network virtualization and fine-grained QoS management by using the common overlay for network virtualization and the OpenFlow overlay to support fine-grained QoS management. This approach has the ability to selectively provide fine-grained QoS support to the needy slices or the flows in a slice to effectively use the switch resources.

As shown in Figure 1, our layered overlay constructs multiple OpenFlow overlays on top of a common overlay that is used to virtualize a network for each cloud tenant. The OpenFlow overlays on top of the common overlay isolate the flows with different QoS requirements within each virtual network. This mechanism is based on one-to-one matching of a virtual network to a tunnel and then a tunnel to a slice. It can be realized in an enhanced OpenFlow switch supporting with three features: hybrid data forwarding (conventional switching and OpenFlow based flow switching), tunneling, and resource slicing. The first feature can be simply provided by using OpenFlow enabled switches; the second feature, tunneling, can be explicitly supported by extending the OpenFlow switch specification (version 1.3) to avoid the limited addressing space or vendor specific issues existing in current approaches [5]. Specifically, Generic Routing Encapsulation (GRE) tunneling can be enabled at the switch level through adding GRE en/decode module into the switch data plane and extending the OpenFlow switch specification. Specically, we can add OXM OF GRE TUNNEL field into the current flow match fields to support GRE package matching and add GRE Key/Header Push and Pop actions into current OpenFlow actions to facilitate GRE encoding/decoding in each switch. The third feature can be supported by devolving the centralized slice controller such as FlowVisor [1] controller to a VN slicer module within a switch.

Using this realization, the GRE header stack, the key of GRE, and the slice QoS level can be identified firstly, and then the switch can decide to use the conventional switching for those flows that only need best effort QoS without flow matching and save the flow table and process resources for the flows with more specific QoS requirements. In this way, it supports fine-grained QoS management for the selected flows without sacrificing scalability to support a large number of applications. This approach fits the cloud virtual network environment, in which a large number of tenants physically share logically isolated network infrastructure through a virtual networking mechanism and a large number of the applications with varied QoS requirements are running on top of it.

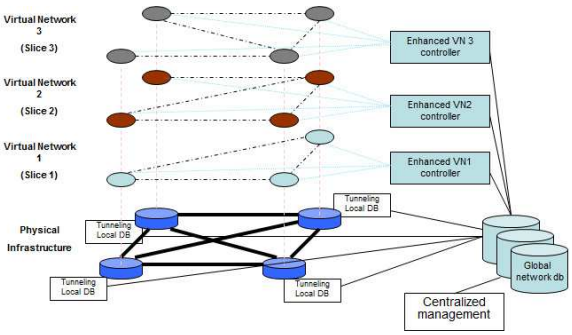

Fig. 2. The DFVisor network virtualization platform architecture.

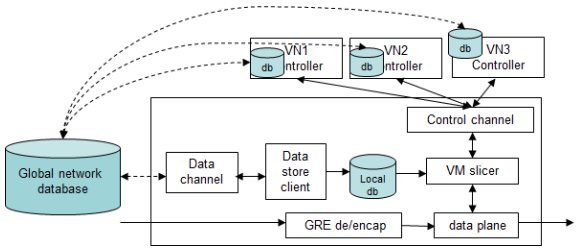

Fig. 3. Enhanced OpenFlow enabled switch structure.

III. The DFVisor Network Virtualization Platform

As shown in Figure 2, our DFVisor consists of three main components: the enhanced OpenFlow enabled switches, the enhanced VN controllers, and the distributed synchronized two-level database system. The layered overlay mechanism is implemented in the enhanced OpenFlow enabled switches by adding a local VN slicer and tunneling module. The enhanced VN controller is a normal switch controller for a virtual network with enhanced OpenFlow protocol to support the tunneling. The distributed synchronized two-level database system consists of a global database and multiple local databases in switches and VN controllers. It stores the network configuration and information and and can be implemented in Apache Zookeeper by taking the advantage of its watch service, which basically allows each local database to put watchers on the global database and receive a watcher notice when the data in the global database gets updated. Therefore, the local database can aware the data changed in the global database and take actions to keep the data synchronization. This two-level database system also creates a dedicated data channel, where the network information predefined or -stored in the global database such as network configuration, topologies, QoS policies, and network statistics can be delivered to each switch and VN controller without invoking any other protocols. Therefore, each VN controller can pre-load its slice and flowspaces information from the global database and construct the flow entry for the traffic in each flowspace within a virtual network (slice) without waiting for a switch to generate a packet-in message when a new flow received; and each VN controller also can periodically updates the network statistics through the synchronization from the global database, to which the switch collects the network statistics and send them periodically. In this way, our proposed DFVisor platform maintains a synchronized global and local network view, which not only simplifies the network configuration and management but also facilitates a push-based flow setup and network statistics collecting mechanism without modifying the current OpenFlow protocol using a dedicated data channel. In this way, our proposed DFVisor platform advances the state of the arts by improving the system scalability limited by problems (2), (3), and (4).

Figure 3 shows the logical structure of the enhanced OpenFlow switch, where a local VN slicer module is added to manage the VN slices through a flow control channel. The switch also maintains a local database, which stores the local network configuration and statistics to facilitate a fast data processing. This local database is also connected with the global database through a database client and a dedicated data channel, which enables data synchronization in both directions. Meanwhile, each VN controller also maintains a local database, which is synchronized with the global database through its global client using the synchronization mechanism presented above. In this way, our DFVisor forms a fully distributed virtual network architecture and solves the six major scalability problems in current resource slicing approaches. It changes the situation that only pull based flow setup and statistic gathering mechanism is supported in current OpenFlow switch specifications and protocols, and facilitates a push-based mechanism without modifying current OpenFlow switch specification and protocol for flow setup and statistic collecting to reduce the flow setup and statistic gathering latency; it also increases the real flow control bandwidth by using another dedicated data channel for network configuration and statistics, such that the scalability issue caused by problems (2), (3), and (4) can be overcome.

Our DFVisor is an end to end network virtualization solution. When a flow initiated by an application inside a virtual machine comes to its access virtual switch, the flow gets GRE encapsulation and is matched to a slice whose configuration and topology have been defined in the global database and synchronized to the related local databases; the tunneled flow is then sent to its next hop according to the virtual network topology. At each hop the flow follows the same routine: de-capsulating, flow matching, and encapsulating until it hits the end virtual switch, where the flow is finally decapsulated and sent to the destination machine. Our DFVisor is a fully distributed architecture that does not add any new single point of failure. It is scalable and flexible. The fine-grained QoS management can be developed as a module on top of a normal switch controller, and the whole system can be implemented based on open source network building blocks.

IV. Issues and Research Directions

Our work presented in this paper is based on our preliminary research. Three major issues remain in terms of how DFVisor performs and how it is best implemented, which will be addressed in our ongoing and future research.

The performance of the DFVisor can be highly affected by the performance of the enhanced OpenFlow enabled switch. The enhanced Openflow switches have the overhead caused by the VN slicer module and the tunnel module. Since the current centralized FlowVisor controller is very lightweight [4], a local VN slicer that only deals with the flows of a virtual network should not impose much overhead on the CPU of a switch. However, finding a way to reduce the resource consumption of the VN slicer control module without sacrificing its performance is the main goal in its current implementation. Developing some mechanism to provision the virtual network while balancing the traffic for each switch is the first research problem for the future.

The DFVisor is developed based on a distributed synchronized two-level database system, which can be implemented on the Apache Zookeeper and its watch service can be used for the two level databases synchronization. This watch based synchronization mechanism creates latency for database synchronization. It may impose longer latency for a switch to receive a flow entry and a VN controller to update the network statistics. However, they are push-based, which means that all the flow entries and statistics are pre-loaded. As long as they can be completed before actually needed, the slightly longer latency shouldn't impact the system performance and scalability. The synchronization between the databases may also increase the network traffic, but we can solve it by forming a dedicated network to manage database synchronization. With the growing of the virtual network scalability, the information inside the global database and the number of watchers added on the global database can grow huge, it creates a big overhead for the global database. However, Apache Zookeeper itself is a clustered structure, adding more physical nodes to this cluster should solve this problem. Analyzing the synchronization mechanism and evaluating its performance is the second research problem for the future.

DFVisor is designed for a large-scale cloud network, but it is difficulty to measure its scalability order since building a large-scale network test bed in a lab is hardly practical. We plan to use network emulator: MININET to emulate a large-scale network for DFVisor prototyping, and use open source network simulator: NS-3 for DFVisor performance and scalability evaluation. Designing test scenarios with considering the variety in network structures, performance parameters, and the risky situation caused by un-synchronized databases, enhancing the NS-3 simulator to enable the scalability order testing is the third research problem for the future.

V. Related Work

Our layered overlay is close to the vertical forward mechanism introduced in [5]. Both concepts use OpenFlow and address the same issue that the current OpenFlow specification has not standardized the approach for tunneling. However, the vertical forward is used to facilitate the data migration between network layers in a telecom network domain while our layered overlay is used to isolate the flows among overlays for the Internet domain. Other solutions that support tunneling at a switch either lack scalability or are vendor specific [6]. FlowVisor's limitation in functionality has been improved by AdVisor[7] and VeRTIGO[8], but the scalability issue is left unsolved. Our DFVisor uses the same resource slicing mechanism in a distributed way and employs the layered overlay mechanism to improve the scalability. Our DFVisor platform is similar to DIAM [9] in using local intelligent control for each switch to improve scalability and avoid single points of failure, but DIAM is focused on network management while our DFVisor targets an end to end network virtualization solution. Midokura[10] and Nicira also provide similar distributed network virtualization solutions, but Midokura uses a common overlay technique for none OpenFlow network fabrics while Nicira uses layer-2 overlay based on VLAN with clustered control plane and a lightweight control agent for each switch.

VI. Conclusions

This paper has addressed the issue that current cloud network virtualization techniques lack capability to provide fine-grained QoS management without sacrificing scalability by introducing a layered overlay concept and a new network virtualization platform called DFVisor based on the enhanced OpenFlow switch structure and a distributed synchronized two-level database system. The proposed DFVisor network virtualization platform nicely solves the six problems that cause the scalability issue. We have introduced the layered overlay mechanism, and present the DFVisor with a push-based flow setup and statistics gathering mechanism. We have explained how the six issues can be solved by using layered overlay and the distributed synchronized two-level database system in DFVisor platform. The potential issues in DFVisor implementation and evaluation are also discussed.

References

[1] R. Sherwood, et al. "Flowvisor: A network virtualization layer." OpenFlow Switch Consortium, Tech. Rep (2009).

[2] Curtis, Andrew R., et al. "DevoFlow: scaling flow management for high-performance networks." *In ACM SIGCOMM Computer Communication Review, vol. 41, no. 4*, pp. 254-265. ACM, 2011.

[3] Apache. "Apache ZooKeeper Programmers Guide", Internet: http://zookeeper.apache.org/doc/zookeeperProgrammers.html.

[4] R. Sherwood, G. Gibb, et al. "Can the Production Network Be the Test Bed?" *in Proceedings of the10th USENIX Conference on Operating Systems Design and Implementation (OSDI 10)*, pp.114.

[5] G. Hampel, M. Steiner, T. Bu, "Applying software-defined networking to the telecom domain", *In Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on* (pp. 133-138) IEEE.

[6] J. Kempf, B. Johansson, S. Pettersson, et al. "Moving the mobile evolved packet core to the cloud", *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on. IEEE, 2012* pp: 784-791.

[7] Salvadori, Elio, et al. "Generalizing virtual network topologies in OpenFlow-based networks", *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011.

[8] Doriguzzi Corin, R., et al. "VeRTIGO: network virtualization and beyond", *Software Defined Networking (EWSDN), 2012 European Workshop on.* IEEE, 2012.

[9] Banjar, Ameen, et al. "DAIM: a Mechanism to Distribute Control Functions within OpenFlow Switches", *Journal of Networks 9*, no. 01 (2014): 1-9.

[10] Akane Matsuo, "Introduce to Network Virtualization for IaaS Clouds", *LinuxCon Japan 2013*, Japan, May 31, 2013. Internet: http://events.linuxfoundation.org/sites/events/files/cojp13 matsuo0.pdf.