OpenVirteX: A Network Hypervisor

Ali Al-Shabibi*, Marc De Leenheer*, Matteo Gerola†, Ayaka Koshibe*, William Snow*, Guru Parulkar*
*Open Networking Laboratory, Menlo Park, CA 94025, USA
†CREATE-NET, Via alla Cascata 56/D Povo, 38123 Trento, Italy

ABSTRACT

Virtualized environments have been around for a long time, but until recently these have focused solely on compute virtualization and left the network behind. Network virtualization enables multiple tenants to share the same physical infrastructure by decoupling the physical network from the virtual network. In this paper, we introduce OpenVirteX, a Network Virtualization Platform which provides virtual Software Defined Networks (vSDNs). Each vSDN is customizable in terms of topology as well as addressing scheme.

Introduction. Techniques for virtualizing network infrastructure have lagged behind that of compute resource virtualization. This has hampered the ability of operators to provide full virtualization that extends down to the network. In the ideal case, operators would be able to provide networks whose topologies, management schemes, and use cases are under the full control of their tenants. Realization of this scheme requires the ability to construct virtual networks that are not only logically isolated, but also have fully configurable topologies and properties akin to virtual machines, e.g., can be created, destroyed, snapshotted, and migrated on demand. Such a mechanism would provide operators with the ability to give a tenant the illusion that they have full control of topology, addressing, and management of the infrastructure despite sharing it with other tenants.

Here we present OpenVirteX, a network hypervisor that enables operators to provide this form of network virtualization to their customers. We leverage the growing focus on Software Defined Networks (SDN) amongst infrastructure providers aiming to simplify and add flexibility to the process of provisioning networks. In specific, OpenVirteX builds on OpenFlow [1] as the SDN medium, and our experiences with FlowVisor [2] for design. Like FlowVisor, OpenVirteX functions as a proxy within the control channel, presenting OpenFlow networks to tenants, while controlling the underlying physical infrastructure via the southbound OpenFlow interface. Figure 1 depicts this architecture. By exposing OpenFlow networks, OpenVirteX allows tenants to use their own network OS (NOS) to control the network resources corresponding to their virtual network. In other words, OpenVirteX creates multiple virtual software defined networks out of one. Unlike FlowVisor, which simply slices the entire flowspace amongst the tenants, OpenVirteX provides each tenant with a fully virtualized network featuring a tenant-specified topology and a full header space.

Motivation. OpenVirteX offers a pragmatic approach to meeting the requirements of both operators and tenants in increasingly common computing situations:

- **Cloud:** OVX facilitates the migration of an enterprise to a Cloud Infrastructure. Each customer can request an exact copy of his/her physical network, with the same address scheme and topology of the current network. The level of abstraction offered by OpenVirteX creates the illusion that the Cloud physical infrastructure fits exactly the client requirements, in terms of isolation, topology and addressing scheme.
- **Provider:** Similarly, in a WAN scenario, virtualization is essential, e.g. to allow multiple enterprises to connect their remote sites. With OVX, this could be done easily, spawning virtual networks that provide inherited traffic isolation, security, and flexibility. Moreover, with OpenVirteX, the tenant can still control how traffic is forwarded inside his network through a NOS and can apply policies (e.g. firewall, load balancer) that in a standard network are completely managed by the ISP.

The remainder of this abstract describes the virtualization approach taken by OpenVirteX, and its current state.

Topology Virtualization. OpenVirteX maintains internal representations of the physical and virtual networks as a collection of software switches, links, and end hosts. The physical topology representation is the basis upon which virtual networks are mapped. Tenants can request a topology by supplying a mapping between the elements in the physical topology and their desired virtual network. Virtual topologies can range from an exact copy or subset of the physical network, over a giant virtual switch that abstracts away all physical switches and links, to any custom topology as described by the tenant. The ability to control topology abstraction allows tenants to simplify their NOS, or operators to implement policies in the network. For example, a tenant may forego loop resolution in their NOS by requesting a loop-free topology, or an operator can require traffic to pass through chosen nodes by modifying the connecting path of a virtual link or between the ports of a giant switch.

The mapping between virtual and physical elements are stored in a global map, but otherwise kept separate. This allows tenant networks to be reconfigured, halted, and re-initialized by simply modifying this mapping, i.e. pointing keys to different values. Finally, since virtual topologies need not correspond to the physical network, resilience can be enabled in the former

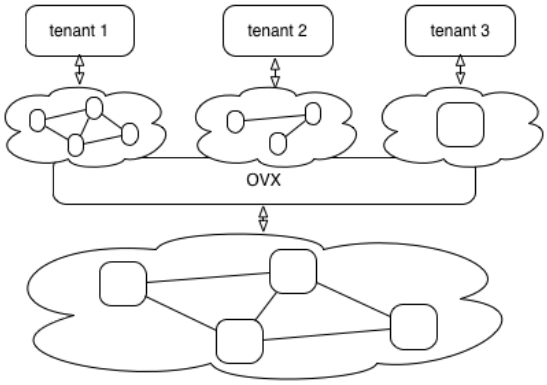

Fig. 1: Network virtualization with OpenVirteX.

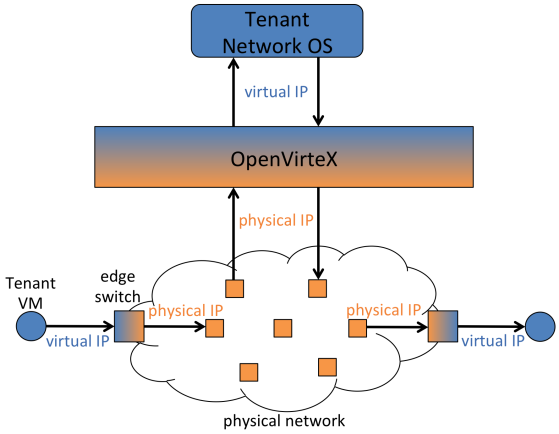

Fig. 2: The address virtualization process.

by allowing tenants to specify multiple backup paths for any virtual links between switches, or within giant switches, between ports.

Address Virtualization. OpenVirteX grants tenants the ability to choose address assignments for their end hosts, allowing multiple, potentially-overlapping IP address blocks to exist in the same physical network. To differentiate hosts, OpenVirteX generates globally unique tenant IDs for each tenant, and for each host, a physical IP addresses that encodes the host's membership using the tenant ID. Collisions of addresses are avoided by installing flow rules to rewrite addresses at the edge switches of the network, from tenant-assigned address to physical IP address at the ingress edge, and vice-versa at the egress edge. The translation process is illustrated in Figure 2. Importantly, this procedure is invisible to a tenant's NOS or hosts, implying that a NOS used to control a virtual network created by OpenVirteX doesn't need to be modified in any way to properly function. In addition to preventing address aliasing in a transparent way, the mapping created by the procedure is used by OpenVirteX to demultiplex northbound control messages so that they reach the correct tenant networks.

Control Function Virtualization. Each virtual network may run its own NOS to program the virtual switches. OpenVirteX makes this possible by mapping various control functions for the virtual network on to the corresponding physical network resources. The translations can get complex, since an operation for one virtual switch may map onto multiple switches and links in the physical network. OpenVirteX leverages its position as proxy that allows it to intercept and manipulate the control packets before they reach their intended destination, which OpenVirteX also determines based on its virtual-physical mapping.

Implementation. We have implemented a preliminary version of OpenVirteX in Java. This version currently supports OpenFlow 1.0, and supports the creation of custom topologies under overlapping IP address spaces, the ability to start and stop virtual networks and their components, and persistent storage using Mongo DB. Backup paths for resilient links and giant switches can be manually configured, or automatically generated with a simple SPF algorithm. A JSONRPC API is provided to enable configuration and monitoring of both tenant and infrastructure networks. This API has been used for an implementation of a GUI, as well as a topology embedder that generates virtual-physical mappings as configurations to OpenVirteX.

Currently on-going work includes the addition of features such as virtual network snapshotting and migration, dynamic configuration of virtual networks, and external and inter-tenant connectivity. In the future topics such as HA and scalability may be explored.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks", in *ACM SIGCOMM Computer Communications Review*, April 2008.

[2] R. Sherwood, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, Can the production network be the testbed?, in Operating systems design and implementation, October 2010.