An SDN-based Network Virtualization Architecture Using LISP

Vina Ermagan, Fabio Maino - Cisco Research {vermagan, fmaino}@cisco.com

The need for advanced network control functions such as traffic Engineering, service insertion and service chaining, multi-tenancy, and tenant isolation are increasingly growing as Network Virtualization becomes the dominant trend in the Data Center and Cloud. The rise of such new requirements has exacerbated the inadequacy and lack of flexibility of today’s network management and control mechanisms especially in the face of rapidly changing virtualized networks. Software-Defined Networking (SDN) has emerged as a general paradigm based on decoupling and refactoring the network management and control logic from the network devices. This refactoring of the network control plane facilitates unifying the elementary functions such as element discovery and state distribution across various network applications. Furthermore, it enables and promotes the design of network management and control logic based on a global network view [6]. Enabling such logically centralized control plane, together with providing open interfaces to the network switches and routers gives the promise of improving the state of the art, by making it easier to extend and introduce new network control functions [7]. Nevertheless, during the past few years, the shift from fully distributed control planes to logically centralized network control hasn’t been without challenges. While centralizing management functions clearly simplifies network management, logically centralized control planes tend to face limitations in scalability. As a result, physical distribution is used as a scaling enabler for such logically centralized control planes. However, research shows that there is a clear trade off between network application performance and application complexity when taking into account this underlying state distribution of the logically centralized control [8]. Consequently we ask the following: how much of the control logic needs to be refactored to achieve the right balance of flexibility and programmability vs. scalability and robustness?

While responding to this question requires extensive research and experimentation, in this talk we will describe how we leverage the Locator/ID separation Protocol (LISP) [4] architecture to implement a network virtualization solution that also provides decoupling of the control plane from the data plane. The core of the LISP architecture lies in its Mapping System, where policies can be defined programmatically, and are fetched and enforced by network switches/routers as flows arrive. We elaborate on how this technology, integrated with a centralized network management framework, can be used to enable new control requirements raised by network virtualization such as multi-tenancy, tenant traffic segregation, and VM mobility, as well as network programmability capabilities such as traffic engineering and service chaining. LISP implementations are available in the open source community including LINUX and FreeBSD.

LISP Overview - LISP is an open protocol that was originally proposed as a framework for decoupling host identity from its location. This decoupling is achieved by defining two separate namespaces: Endpoint Identifiers (EIDs), and Routing Locators (RLOCs). EIDs can be L2 identifiers (MAC addresses) or L3 identifiers (syntactically identical to IP addresses: 32 bit for IPv4 or 128 bit for IPv6) that identify the host attached to the network, while RLOCs are IPv4 or IPv6 addresses that represent network attachment points of these hosts. Host applications bind to the host’s EIDs for layer 4 transport connections. Packets with headers in the EID namespace are encapsulated in a second header from the RLOC space, and are routed towards the destination. Upon reception at the destination, the LISP header is removed before delivering the packet to the end-host. LISP introduces special gateway routers called Tunnel Routers (xTR) that perform the LISP encapsulation, and decapsulation at a LISP site’s ingress and egress points. Separating the host identity from its locator enables seamless endpoint mobility by allowing the applications to bind to a permanent address, the host’s EID. When the host’s location changes, LISP tunnel router will encapsulate the packets towards the new RLOC, preserving the transport connection from breaking. The L2-in-L3 LISP [11] encapsulation header is the same as the VXLAN encapsulation. However LISP promotes the use of L3-in-L3 encapsulation, which offers better scalability and autonomy. For a more detailed description of LISP please refer to [4].

LISP and SDN - LISP provides a flexible map-and-encap framework that can be used for overlay network applications, including data center network virtualization. The LISP framework decouples network control plane from the forwarding plane by providing: (1) a data plane that specifies how the virtualized network domains in the EID space are encapsulated in RLOCs from the underlying physical network, and (2) a control plane that stores the mapping of the EID space to RLOCs and associated policies (Mapping System) and specifies the interfaces and protocol by which elements from the forwarding plane, including the network virtualization edge and/or core devices, can use to query/update this mapping and policy information. The LISP control plane enables network programmability by providing two main components: (1) The mapping system which is a distributed, highly available database that stores the virtual-to-physical mappings and associated policies, and (2) a light weight control
The LISP mapping system enables network virtualization: a single mapping infrastructure can run multiple instances of the mapping database each identified by a 24 bit Instance ID enabling multi-tenancy. The mapping system stores mappings of virtual end points or subnets (EIDs or EID prefixes) to the underlying physical network, along with associated policies. The LISP Canonical Address Format (LCAF) [9] defines an extensible format for including policies associated with these mappings. Examples include list of RLOCs with associated weight and priorities for load balancing, TOS and flow label, geo-locations, etc. It also defines an Explicit Locator Path (ELP) allowing for specifying a sequence of network hops (virtual or physical) that the specified flow must traverse before being forwarded to the final destination. The ELP can be used to define traffic engineering and service chaining policies. As the flow arrives at each hop in an ELP, the local control plane will query the mapping system to fetch the next hop for the flow, including any associated policies, and re-encapsulates the flow toward the next hop in the path. Depending on the requirements of every network deployment, the granularity of the mappings and policies associated can vary. This allows for tuning the control plane functionality between the two components of the control plane: (1) the application that defines and modifies the mapping and policy information in the mapping system, and (2) the light weight control plane on each network device that fetches and enforces the forwarding policies in the local switch. This tuning capability facilitates achieving the desired balance of flexibility and programmability vs. scalability and robustness for a particular deployment.

The LISP architecture has a modular design [10] that allows the use of different distributed storage architectures, provided that the interface to the mapping system remains the same. This allows for deploying different mapping databases that fit the requirements of different network deployments. Among currently proposed and deployed databases are BGP-based overlays [5], hierarchical databases similar to DNS, and DHTs.

LISP has been operational for more than 4 years in the context of WAN deployments, first on an experimental basis, which evolved into today’s LISP Beta Network that includes over 140 LISP sites in 25 countries, operating in both IPv4 and IPv6 EID and RLOC space. Open source implementations of LISP are available in FreeBSD [3] and LINUX [2]. LISP is also supported in the latest version of Open vSwitch [1].

References