NeMo – An Application's interface to Intent Based Networks

What's NeMo – A *Net*work *Mo*deling Language?

NeMo is a transaction based North Bound (NB) API which allows applications to use intent-based policy to create virtual networks comprised of nodes with policy-controlled flows. Intent base policy is prescriptive (ñgo to the storeò) rather than descriptive (ñfollow this route to the storeò) leaving the details to the network. NeMoôs NB API connects the application to a controller and operates using 10 commands which include: 4 basic network commands (Node, Link, Flow, Policy) and 6 basic controller communication commands (connect, disconnect, transact, commit, notification, query). NeMo sends these 10 commands via the REST protocol to exchange the commands with the controller.

Why NeMo?

Software Defined Networking (SDN) and Network Function Virtualization (NFV) are moving the IT world from a network-centric view to an application view. Google considers the datacenter to be comprised of compute devices, storage devices, and networks. Applications running on the Google Cloud must be rewritten to run within this Cloud environment that takes care of placing applications on compute devices that have the appropriate amount of storage and network connections. NeMo provides a simple NB API which gives the application the power to setup and take down virtual networks between virtual nodes.

NeMo's Virtualization

NeMo uses the applicationôs view of the compute, storage, and network which allows the application to set any grouping of compute, storage, or network as a virtual ñnodeò. This flexible viewpoint allows the application to decide what constitutes a compute node and what constitutes a ñlinkò and a ñflowò. From the applicationôs viewpoint, it intends to connect two nodes. It does not matter to the application if the node is a single VM or a cluster of interconnected compute and storage devices with many network connections. NeMoôs NB API hides this complexity, making the applicationôs commands prescriptive and simple. A description of our demo will demonstrate how NeMoôs API provides this simple but powerful interface.

The NeMo Project has created an open-source demo of NeMoôs NB API that allows applications to communicate over layers of virtual devices. The NeMo demo runs on an Oracle VM Virtual Box (www.virtualbox.org) that uses the Mininet VM as its ñactual nodeò network of nodes and links. The NeMo demo uses 3 virtual machines as 3 controllers. Figure 1 shows this basic topology.

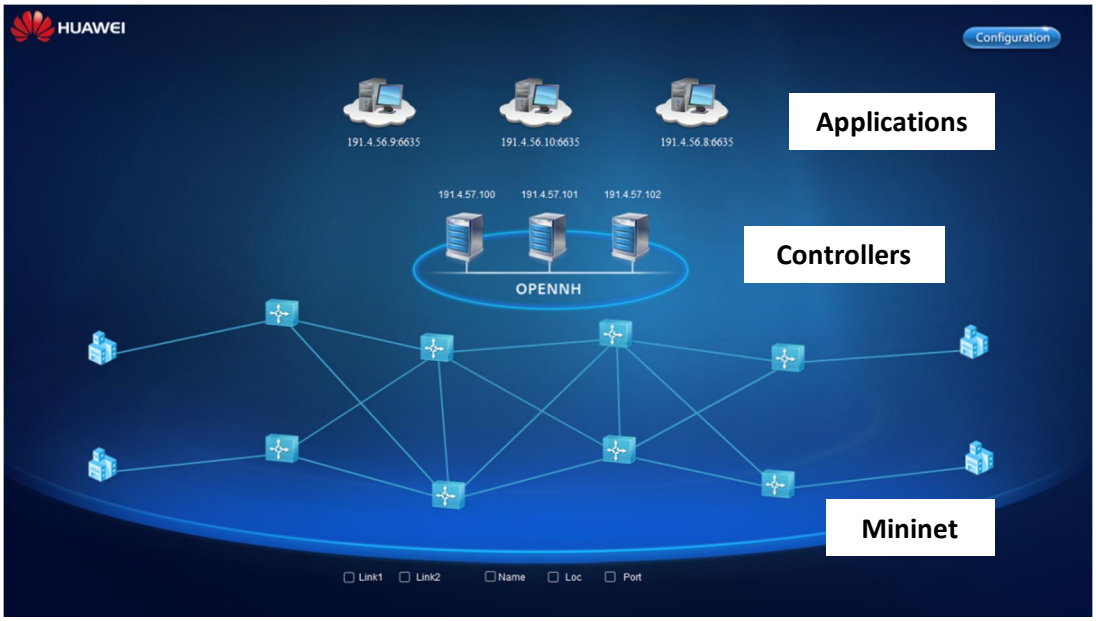

Figure 1 NeMo Demo topology running inside of an Oracle Virtual Box

An Application exchanges NeMo commands using REST Protocol to controller (OpenNH) to instruct the controller to set up a virtual network of nodes and links with flow policy to control the data flows across the network links. The applicationôs logical topology is shown in Figure 2.

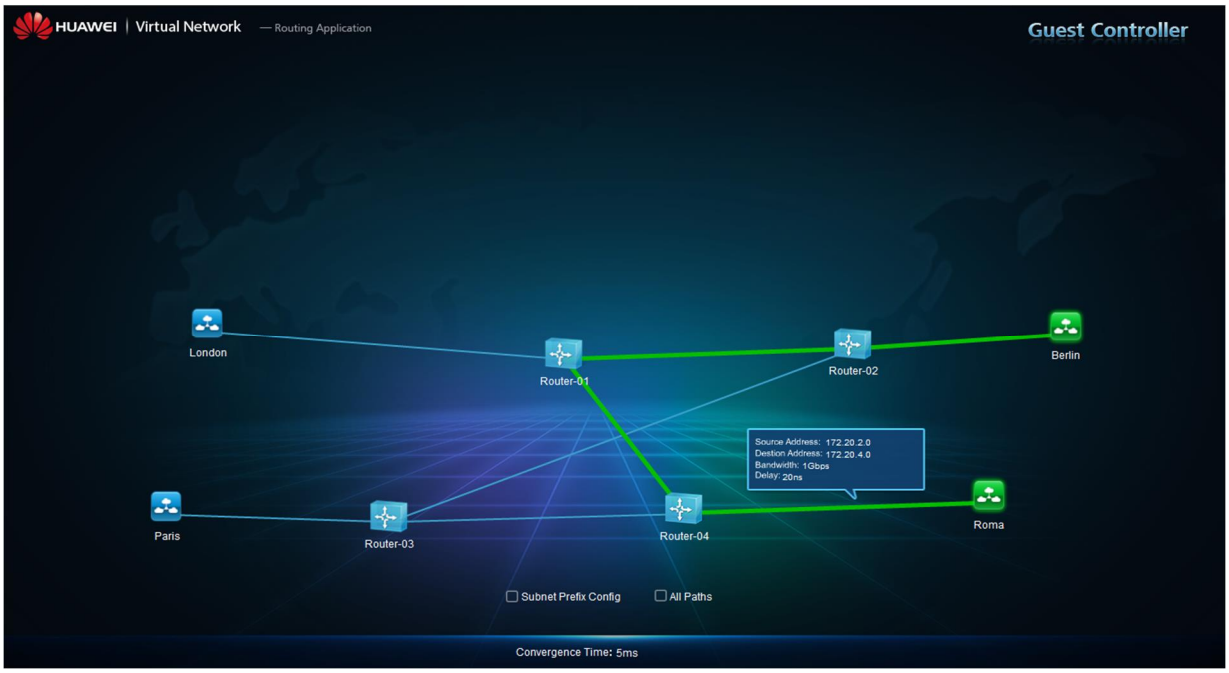

Figure 2 Application's logical topology in NeMo Demo

In the demonstration, the virtual controller (OpenNH) uses OpenFlow to set-up the application's virtual network with nodes and links on the top of the Mininet's VM-network and send data flow policy to each node. Figure 3 shows the mapping of the application's logical topology to the Mininet VM network. NeMo's NB API prescriptive language leaves the details of the underlying network to the network controller. This network controller could create the network out of physical nodes, virtual nodes, or a combination of both.

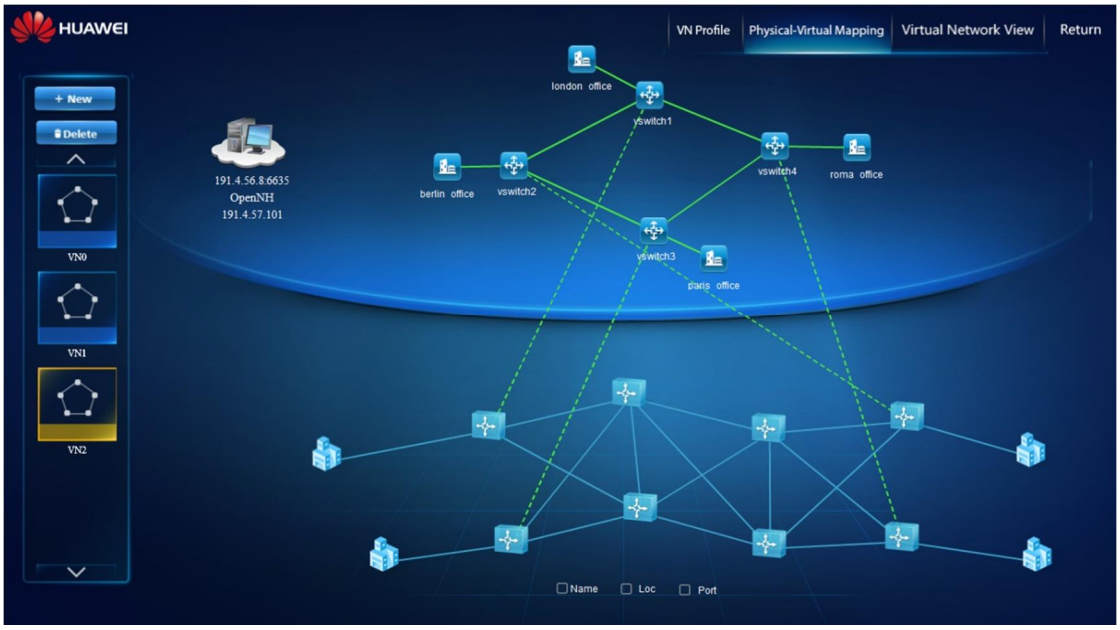

Figure 3 - NeMo Virtual Network overlaid on Mininet

NeMo Commands for this topology

NeMo versus Group Policy in Open Daylight

Open Daylight has a group-based policy project that defines an ñapplication-centricò model that tries to abstract application policy from details of network policy. Two groups of endpoint systems communicate across a shared communication path governed by a contract on the exchange which one system (EGP 1) and the second system (EGP 2) consumes. The contract consists of policy rules which govern flows specified as match-action pairs. (Details on this project can be found at: https://wiki.opendaylight.org/view/Group_Policy:Main).

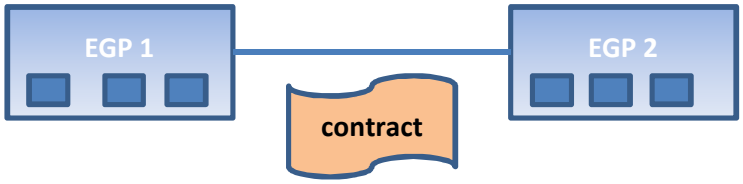

Figure 4 – Open Daylight Group-based Policy concepts

The NeMo project group applauds the Group-Policy project for its adoption of the Internet-based framework, and the prescriptive rather than descriptive language. The NeMo Project group believes that the Group-Policy focuses on the storage and retrieval of policy data leveraging concepts found in RFC3060 and RFC3460. However, we feel the Group-Policy Projectôs application can be made even simpler. We are excited that both groups continued their demonstration and prototype work so that we can find a prescriptive intent based NB interface that is simple and powerful. Applications need a simple transaction oriented interface (much like SQL had for databases) for network set-up.

NeMo vs OpFlex

The OpFlex protocol (draft-smith-opflex-00) is another intent-based protocol which declares the policy infrastructure and allows controls to make it happen. The NeMo team is pleased to see policy repository work that considers centralized methods of policy storage. The OpFlex protocol stores policies in a central policy repository (PR) and distributes these policies to policy elements (PE) where the policies are enforced. The OpFlex control protocol allows bidirectional communication between the central policy repository (PR) and the PE so that the policy, events, statistics and faults can go both directions. OpFlex re-invents many of the same concepts of centralized policy control point (PCP) and policy enforcement point (PEP) that the COPS protocol (RFC2748, RFC3084, and RFC4261) contained. This work is an exciting placement outgrowth of the intent-based policy for centralized storage, but a consideration of virtualized centralized storage (with physically logical points) may also need to be considered.

NeMo's 4 Network Commands in Formal Language

<node_cu> := **Node** <node_id> **Type** <node_type> **LogicNW** <node_id>

[Property {<property_name>: <value>}];

<link_cu> := **Link** <link_id> **EndNodes** <node_type> : <node_id>, <node_type> : <node_id>

[Property{<property_name>: <value>}];

<flow_cu> := **Flow** <flow_id> **Match** {<property_name>: <value>

| **Range** (<value>, <value>) | **List** ({<value>})}

<policy_cu> := Policy <policy_id> ApplyTo <entity_type> : <entity_id>

Priority <integer> [**Condition** {<expression>}]
Action {<action_type> : {<value>}};

A ñNoò in front of each of these commands deletes these commands. An example of this is:

<node_del> := **NoNode** <node_id>;

NeMo's 6 Controller Interaction Commands in Formal Language

!*These commands connect and disconnect the application from the controller.*

<connect_cmd> := **Connect** < conn_id > **Address** <ip_addr> **Port** <integer>

<disconnet_cmd> := **Disconnect** < conn_id >

!*These commands group a set of policy that need to be enacted together.*

<transaction_begin> := **Transaction**

<transaction_end> := **Commit**

! *These commands query or provide notification of policy information*

<query_cmd> := **Query** {< property_name>} **From** <entity_type> | <policy_type>: <UUID>

<notification_cu> := **Notification** <notification_id> **Query** {< property_name>}
From <entity_type> : <entity_id>
Every <integer> **Listener** <callbackfunc>;