NeMo - Network Modeling for Applications
---- An Application API for Intent Driven Networking
Topics

• Why NeMo?
• Status
• State machine
• Demo Description
Why Intent-Driven NeMo?

Application needs Intent-Driven not prescriptive Control

Â Application to state:
   ï A connection between two sites with flows
   ï A service flow with SLA
   ï A customer network service chain

Â Intent Driven: What I want not how to do it
   ï Let network layers figure out how to accomplish intent

Â High level
   ï Yang is low-level specific to device

Applications need a Simple API

Â Request virtual networks through specific nodes with network services at flow rate,

Â When applications can aid control of network, storage, compute – can reach 95% utilization of net, storage, compute

Â NeMo has 3 primitive groups, 15 sentences, and 36 key words
Use Cases Supported

Service Chaining

Virtual WAN with TE
NeMo can enable Multi-service SDN Controller

Multi-Service SDNcontroller

Problem
- It’s hard to support multiple, independently developed SDN applications or services without resource conflicts

State of the Art
- ODL Helium has not solved this problem which prevents competing flow writers that can’t be run simultaneously.
- It is not possible to run e.g. NetVirt and SFC services in the same controller domain.
- Commercial controllers have not solved this problem either

NetVirt + SFC-Ctl

NeMo’s API uses REST/RPC to talk to Nemo Language Engine

- Third Party App
- Third Party Orchestrator
- NEMO Web
- NEMO Language Engine
- VNM Engine
- Multi-Vendor SDN Controller
- Physical Network
NeMo API at App layer rather than ODL Policy Groups

OPL Group Policy

**Purpose:**
- “higher” than neutron policy storage and control

**Benefits:**
- Intent based
- Use PCIM concepts (RFC3060, 3460, 3644) that combine policy rules into policy groups (aka contracts)

**Problem:**
- Only Flow behavior, no create node or specify network service so cannot handle NFV devices or TE channels
- Need Network flows, NFV, SFC, TE plus compute and storage placement

---

Policy Groups architecture

```
  CLI  Heat  Horizon
       /     /     /
      Neutron
       |
     Policy Manager
       |
  Legacy Policy driver  ODL Policy Driver  others
```
Status

Completed: (July – Nov)
- API presented at network forums
- IETF drafts + technical Manual specify language State Machine +
- Proof of Concept demo created

Possible Next Steps:
- Work with Partners on API
- Open Daylight project to integrate NB API + Nemo Engine running over
  - Open Flow with SFC and SFC chaining,
  - I2RS yang modules

We welcome feedback on NEMO, proof of concept demo, and our next steps.
NeMo State Machine

• Top down view
• Entity Model
• Capability Model
• Language primitives: 3 groups, 15 sentences, 36 key words)
Top-Down design
Network Abstraction Model

Network Information

Entity Model
- Link
- Node
  - Forwarding Node
    - Switch
    - Router
    - Transport
  - Processing Node
  - Logic Node

Capability Model
- Data
  - Flow
  - Packet
- Notification
- Policy
- Query

Transaction model
- Connect
- Disconnect
- Transaction
  - commit
Entity Model

- The entity model provides a fundamental abstraction for both basic network objects (such as basic network element, link, and flow) and extended objects (such as firewall, load-balancer, and DPI).

<table>
<thead>
<tr>
<th>Entity Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
</tbody>
</table>

Entities derived from the entity model
Capability Model

- Capability model describes a set of network functions and operations that is opened to the user.
- Two operation modes are defined in the capability model:
  - Synchronous mode: e.g. a creation of virtual network.
  - Asynchronous mode: e.g. port failure notification.

Capabilities derived from the capabilities model
## NEMO Language: Concise and Flexible

### Resource Access

| Entity Model | node | Node/UnNode entity_id Type {FN|PN|LN} Owner node_id Properties key1,value1 |
|--------------|------|-------------------------------------------------|
|              | link | Link/UnLink entity_id Endnodes (node1_id,node2_id) SLA key,value Properties key1,value1 |
|              | flow | Flow/UnFlow entity_id Match/UnMatch key1, value1|Range(value, value) |Mask(value, value) Properties key1,value1 |

### Policy and Event Handling

<table>
<thead>
<tr>
<th>Capability Model</th>
<th>Query</th>
<th>Query key Value {value} From entity_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
<td>Policy/UnPolicy policy_id Appliesto entity_id Condition {expression} Action {“forwardto”</td>
<td>“drop”</td>
</tr>
</tbody>
</table>

| Notification     | Notification entity_id On key Every period RegisterListener callbackfunc |

### Model Definition and Transactions Control

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Connect &lt;conn-id&gt; Address &lt;ip-prefix&gt; Port &lt;integer&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disconnect &lt;conn_id&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node definition</th>
<th>NodeModel &lt;node_type&gt; Property {&lt;data_type&gt;: &lt;property_name&gt;}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link definition</td>
<td>LinkModel &lt;Link_type&gt; Property {&lt;data_type&gt;: &lt;property_name&gt;}</td>
</tr>
<tr>
<td>Action definition</td>
<td>ActionModel &lt;Action_Name&gt; parameter {&lt;data_type&gt;: &lt;property_name&gt;}</td>
</tr>
</tbody>
</table>
NeMo Language Engine
Demos and Documents

Demos – After SDNRG and NFVRG

IETF Drafts:
- [draft-xia-sdnrg-service-description-language-01](#)
- [draft-xia-sdnrg-nemo-language-01](#)

All Project documentation
- Technical Reference
- 5 page summary
- Status of Code
- Presentations
Demo

Virtual sites

Three controllers

Mininet on Vbox
Apps V-Net

Overlay to Mininet
Example of Service Programming by NEMO

App use NEMO language to programming their service:

Flow sitea2siteb  Match srcip:10.0.0.1 dstip:10.0.1.1;
Policy day applyto flow sitea2siteb condition 0800<time<2000 action gothrough {R1,R2,R4};
Policy night applyto flow sitea2siteb condition 2000<time<0800 action gothrough {R1,R3,R4};

Compiler resolver NEMO code to southbound instruction and maintain a state machine for each app.
At daytime go through path1;
Flows in Apps Virtual Network
Q & A