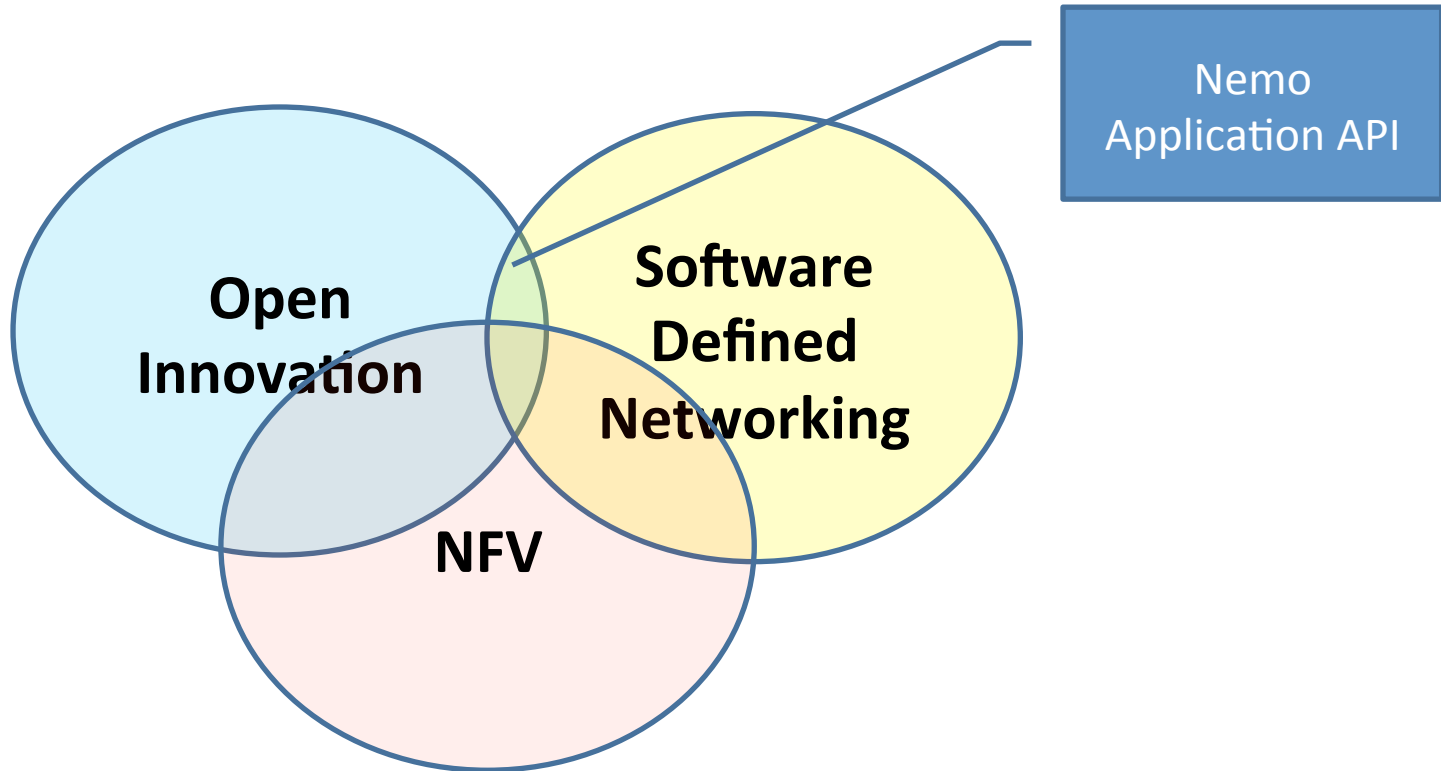




NeMo - Network Modeling for Applications

---- An Application API for Intent Driven Networking

Context for Nemo



Topics

- Why NeMo?
- Status
- State machine
- Demo Description

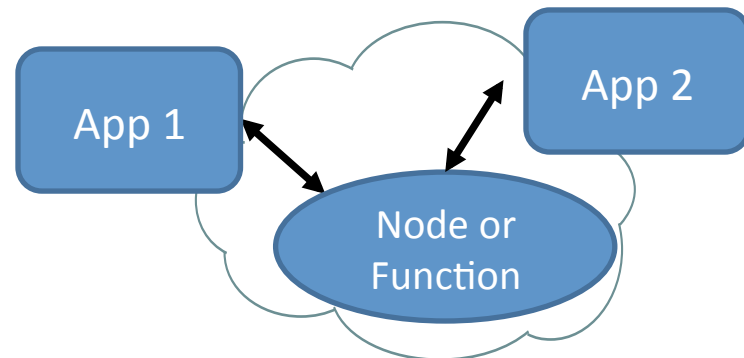
Why Intent-Driven NeMo?

Application needs Intent-Driven not prescriptive Control

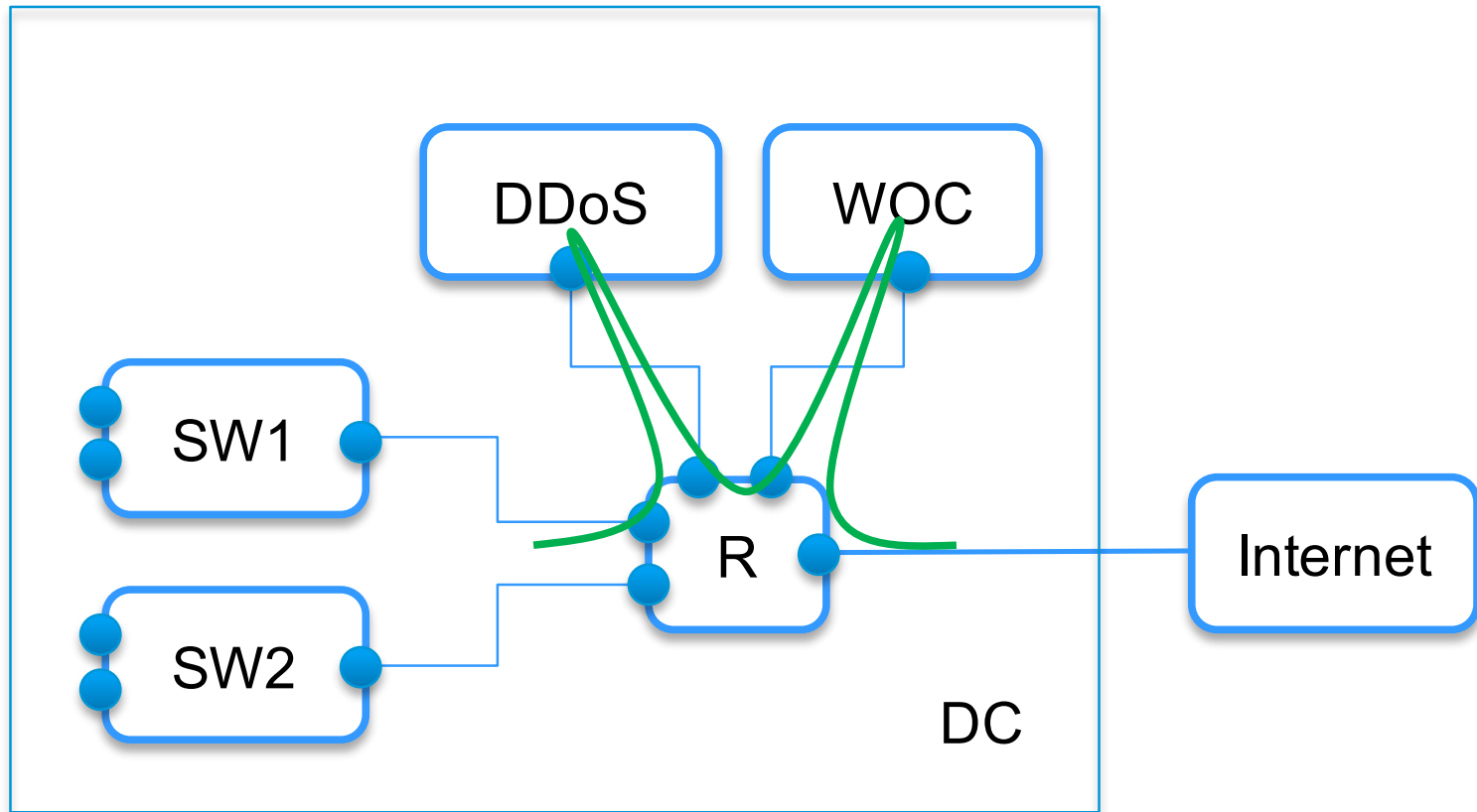
- **Application to state:**
 - A connection between two sites with flows
 - Network Function flow
 - A customer network service chain
- **Intent Driven: What I want not how to do it**
 - Let network layers figure out how to accomplish intent
- **High level**
 - Yang is low-level specific to device

Applications need a Simple API

- Request **Virtual network path through specific functions** with network services at flow rate,
- When applications can aid control of network, storage, compute – can reach 95% utilization of net, storage, compute
- **NeMo** has 3 primitive groups, 15 sentences, and 36 key words



Path through logical functions



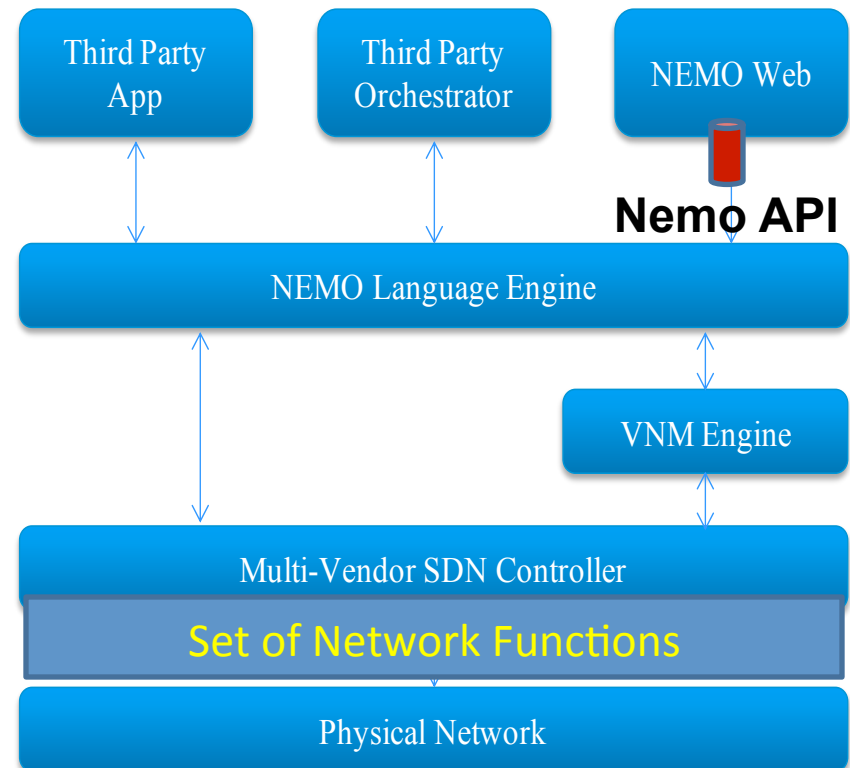
NeMo can enable Multi-service NFV Controller

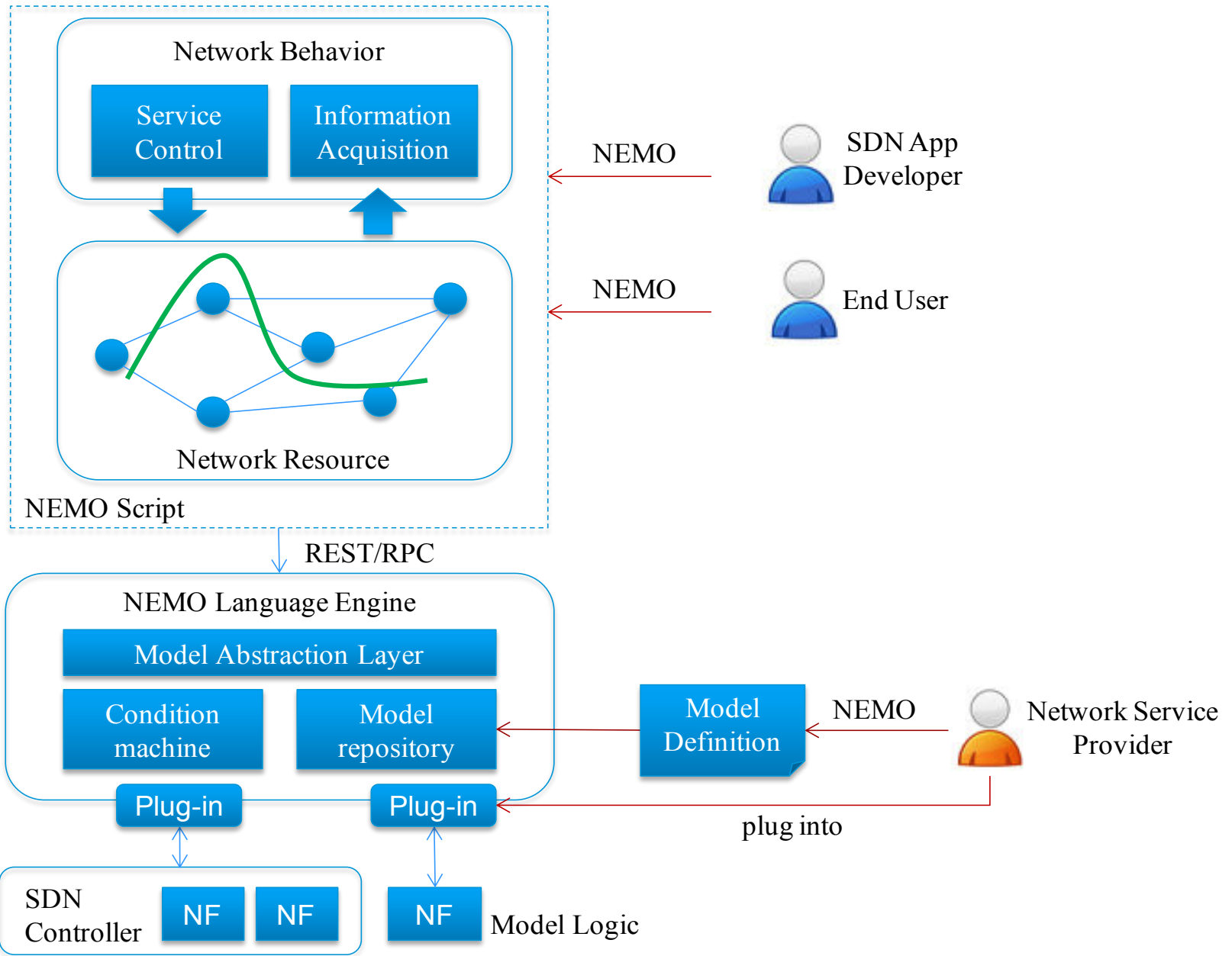
MultiService NFV controller

- **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



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





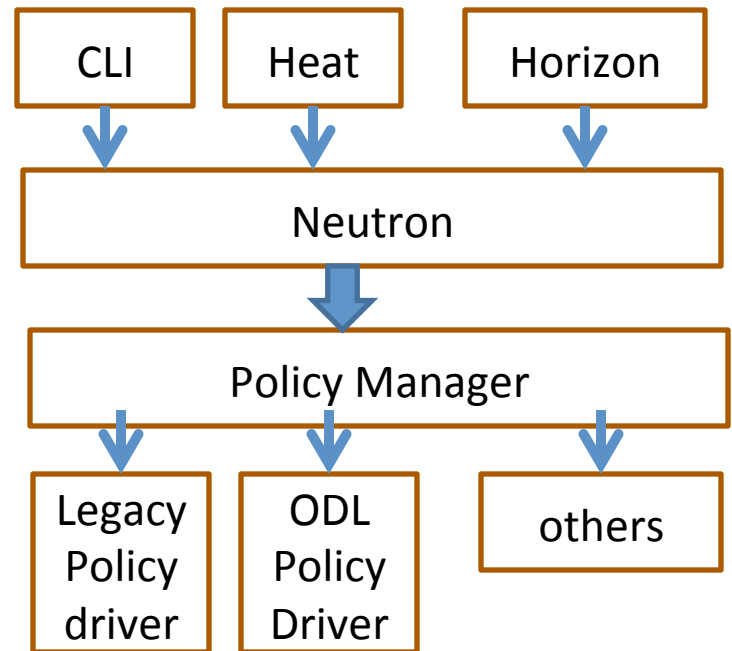
NeMo API at App layer rather than ODL Policy Groups

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



Status

Completed: (July – Nov)

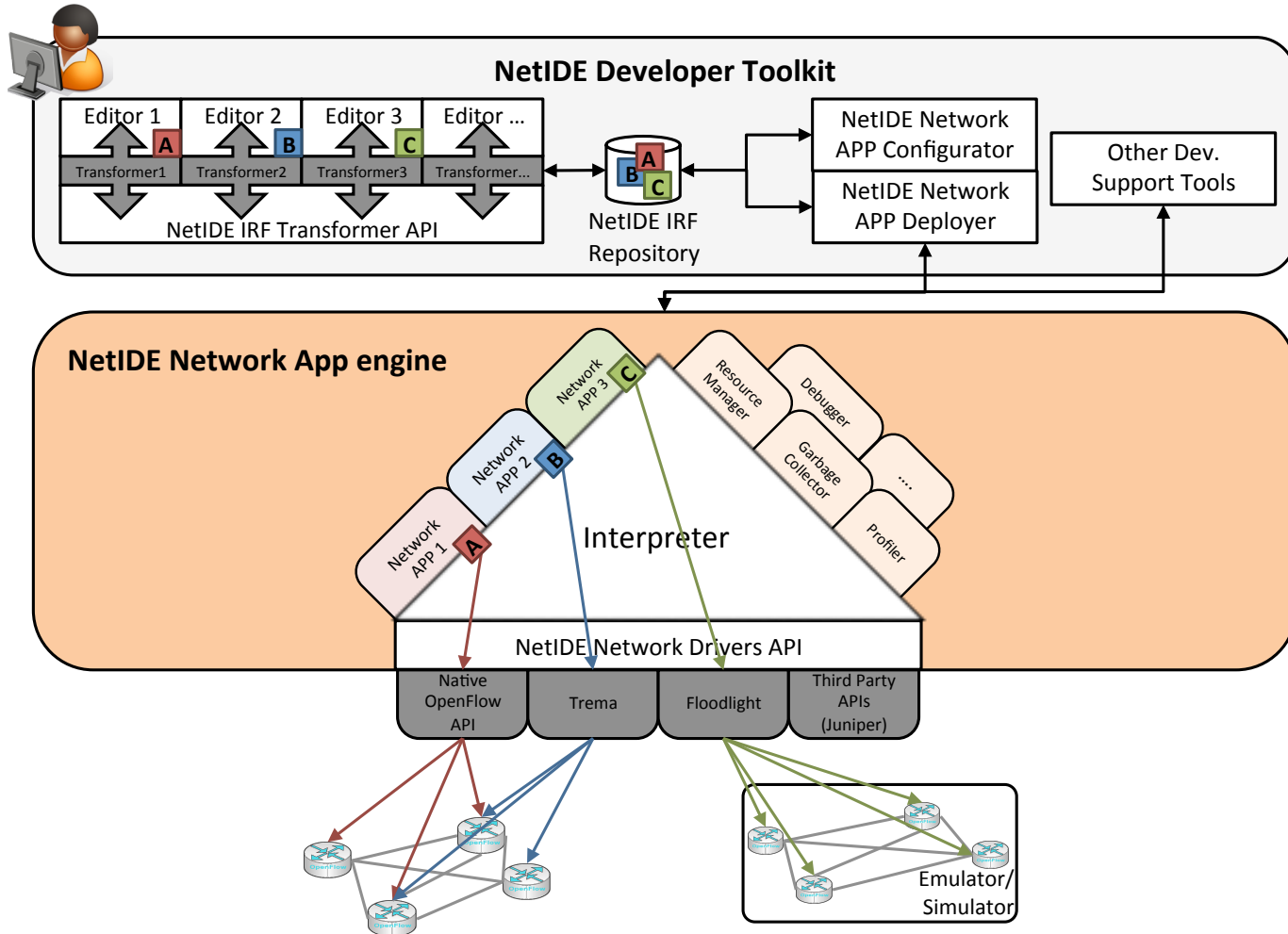
- API presented at network forums
- IETF drafts + technical Manual specify language State Machine
- Proof of Concept demo created

Possible Next Steps:

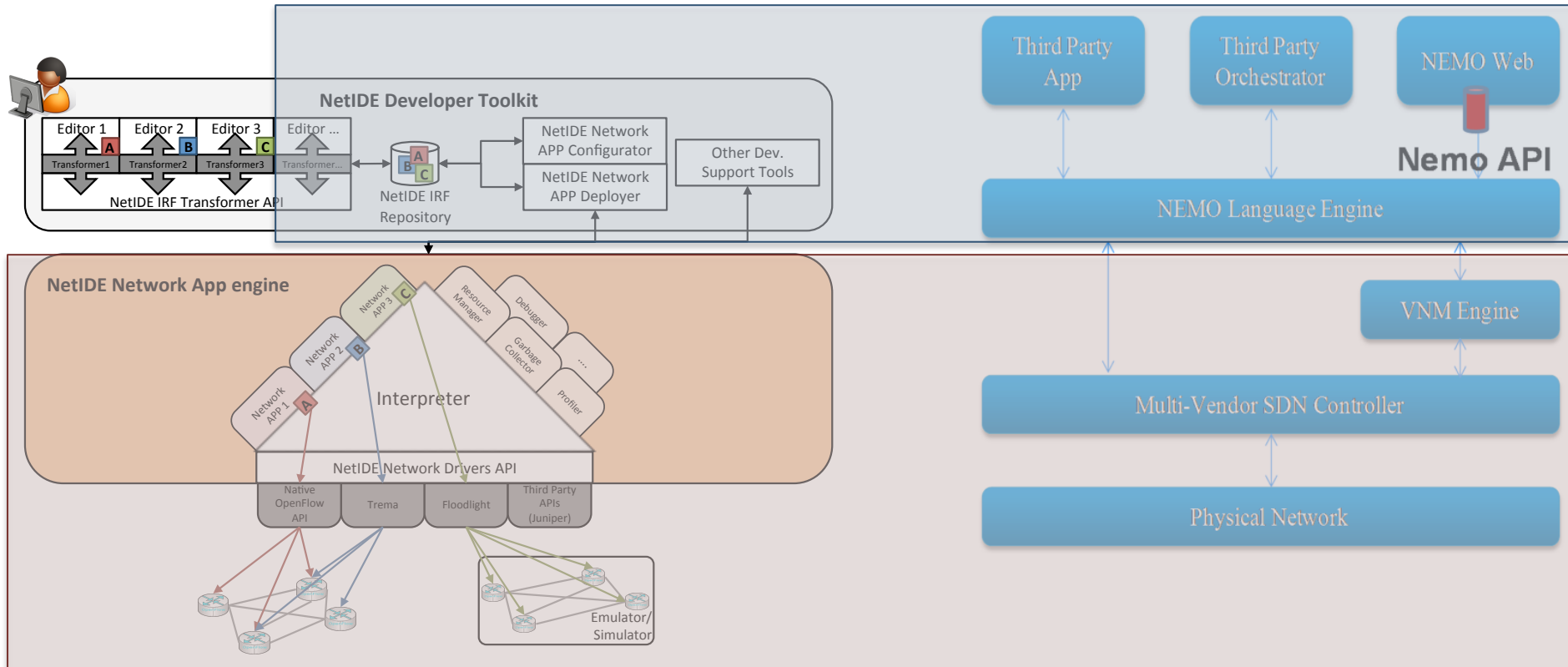
- Work with Telefonica on the Nemo interface to IDE
- Create Open Daylight Project
- Add Security level to API

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

The NetIDE Architecture



Matching NeMo and NetIDE



- NeMo can provide the support for the NetIDE IRF
- NetIDE can act as the NeMo Virtual Network Engine

- * No overlap, clear synergies
- * Align IPR policies

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

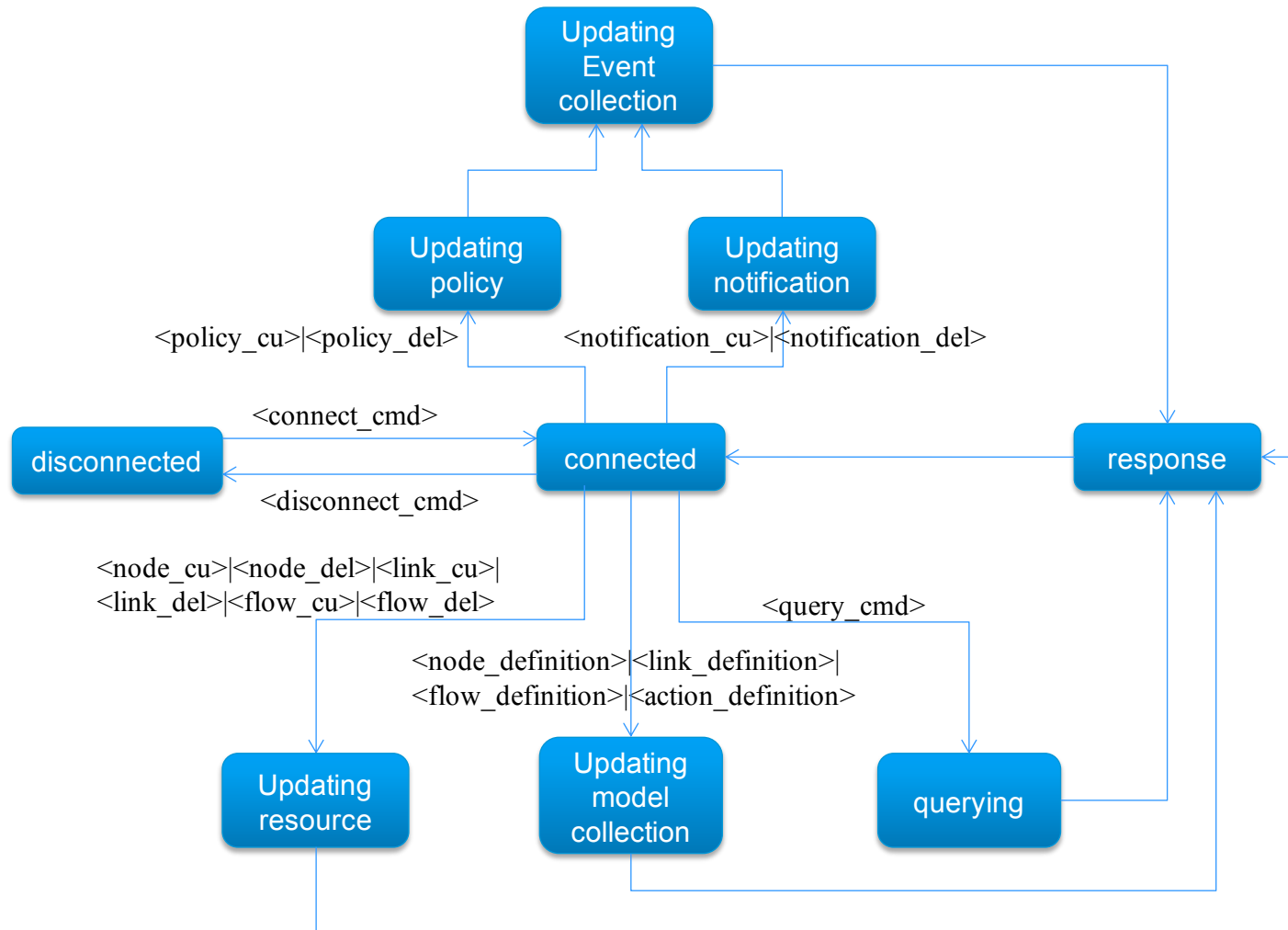
Policy and Event Handling

Capability Model	Query	Query key Value {value} From entity_id
	Policy	Policy/UnPolicy policy_id Appliesto entity_id Condition {expression} Action { "forwardto" "drop" "gothrough" "bypass" "guaranteeSLA" "Set" "Packetout" Node UnNode Link Unlink } Commit / Withdraw
	Notifica-tion	Notification entity_id On key Every period RegisterListener callbackfunc

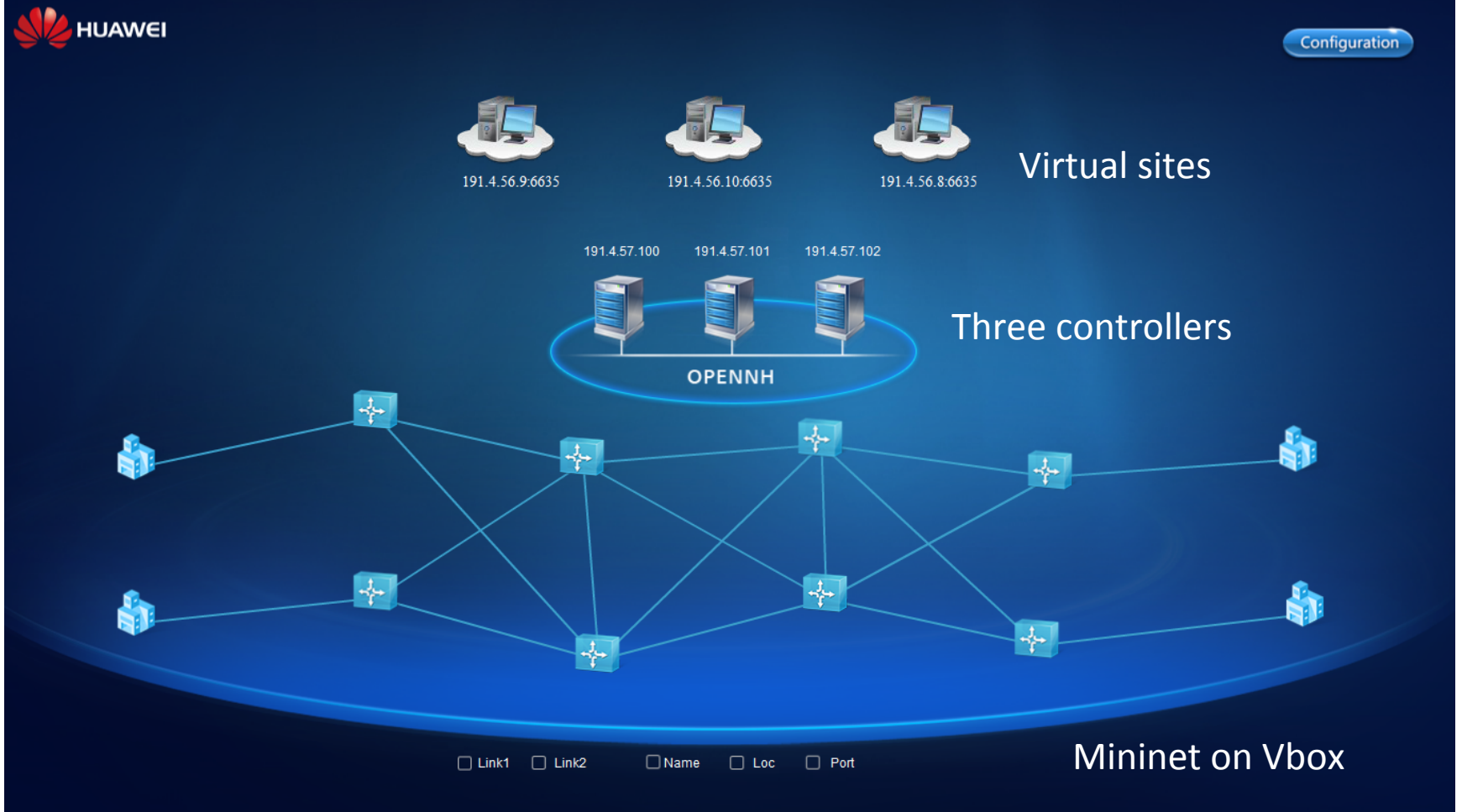
Model Definition and Transactions Control

Connect	Connect <conn-id> Address <ip-prefix> Port <integer>
Transaction	Disconnect <conn_id> Transaction Commit
Node definition	NodeModel <node_type> Property { <data_type> : <property_name> }
Link definition	LinkModel <Link_type> Property { <data_type> : <property_name> }
Action definition	ActionModel <Action_Name> parameter { <data_type> : <property_name> }

NeMo Language Engine



Demo



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

Q & A

Apps V-Net

+ New

Delete

^

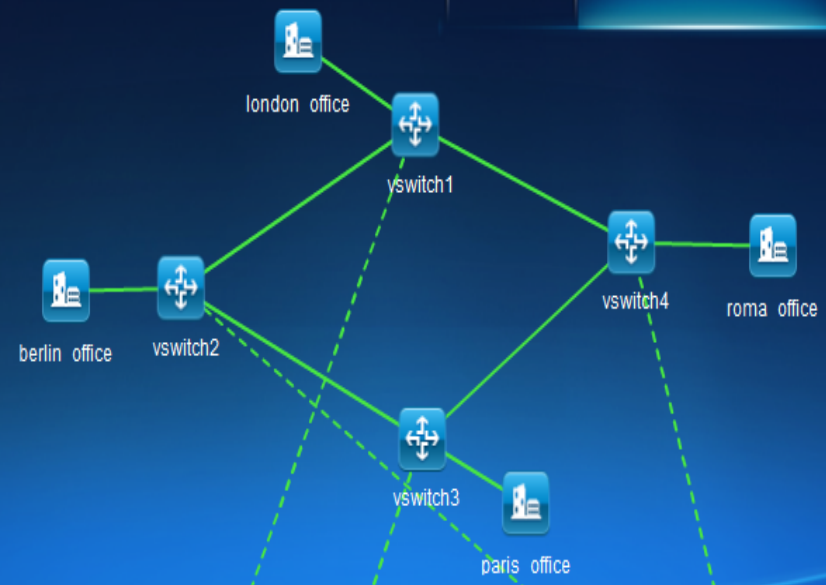
VN0

VN1

VN2

v

191.4.56.8:6635
OpenNH
191.4.57.101



Overlay to Mininet



Name Loc Port

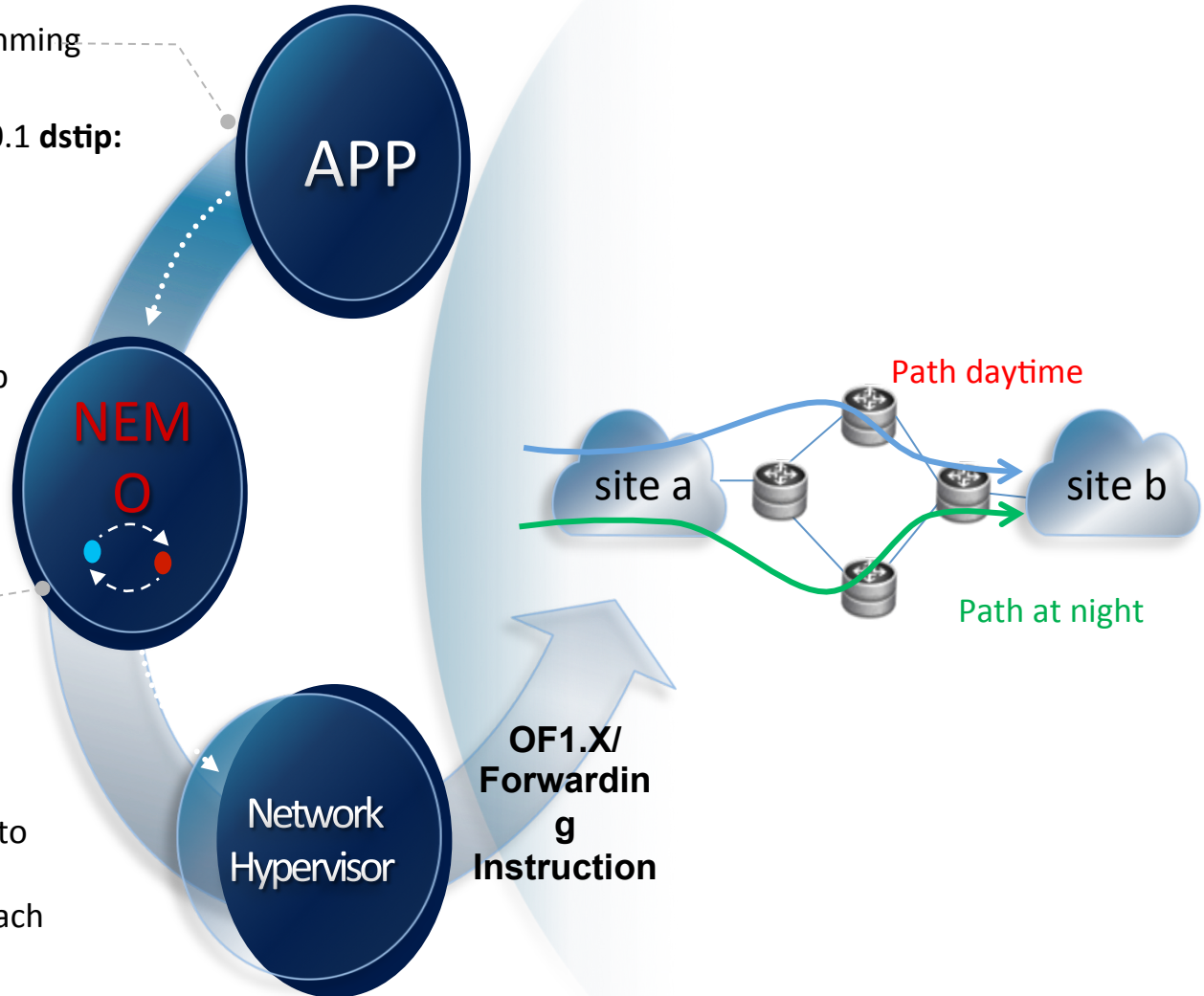
Example of Service Programming by NEMO

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

