Philosophy of Network Operating Systems and Intent APIs

Inder Monga
CTO, ESnet

Technology Exchange,
October 2015
Cleveland, Ohio
Agenda

• Background

• ESnet’s Network Operating System

• What is your Intent?

• Summary and Next Talk!
ESnet SDN investigations, in a nutshell:

2006
- OSCARS, 2006; R&D100 Award 2013
- DOE award, 2015
- RDMA over Ethernet with OF and NEC, 2011
- Zero Configuration Circuits, 2011

2012
- Network Virtualization ‘One Switch’, Nov 2012

2012
- World’s first Transport SDN, Dec 2012
- Software-Defined Exchange @ REANNZ/ESnet, April 2013

2013
- Multi-domain Circuits with NSI, Oct 2013

2013
- Multilayer Transport SDN Demonstration, Oct 2013
- Scalable BGP ‘white-box’ Router, w/ Corsa, Aug 2014

2014
- Dynamic L3 and L2, SDN router + NSI, Sept 2014

Thanks to ESnet’s SDN team
ENOS: A Network Operating System for ESnet Testbed by Eric Pouyoul
Network OS – classic SDN picture

Four Implicit SDN Assumptions

1. Control program configures all network switches
2. Switches relatively homogenous in role/function
3. Switches all use hardware (ASICs) for forwarding
4. Network dataplane is fairly simple (just forwarding)

We were wrong on all of them…..
Confusion between SDN controller and Network Operating System

• The previous picture equates implementation of NOX (or a simple SDN controller like Ryu) to Network Operating System (NOS)

• If one draws an equivalence to a Computer Operating System,
  – a NOS needs more features than **Network State, Topology and OpenFlow message management**

• We started to design an equivalent Network OS + hypervisor (end of 2014)

• ..with the assumption that SDN controller is something we would leverage from open source...
Requirements when designing our Network OS

Philosophy of our SDN approach was to focus on

- **Multiple applications**
  - Support multiple, concurrent applications on the network
  - Agile resource allocation and resource management
- **Network operator**
  - Design with network operations and the current network operator skills
  - CLI access to the entire network
- **Security**
  - Enable different roles: users, applications, network operators, network collaborators, testbeds etc.
- **Multi-domain compatible**
  - NSI and BGP protocol support
- **Legacy internet compatible**
- **Simplicity of use and development**
  - Scripting interface, Python programming model
  - Focus on Intent APIs as research
Computer Operating System

User Programs

User APIs

System Calls

Hardware

- Program control
- I/O
- File System
- Comms
- Error Mgmt
- Resource Mgmt
- Auditing
- Security
Of course, Network Operating System runs on distributed sets of motherboards
High-level architecture of the Network OS

- **Federated applications**
- **ENOS**
- **SDN Controllers**
- **OSCARS**
- **Performance Agents**

**Notional ESnet Network Operating System**
Computer Operating System

User Programs

User APIs

System Calls

Hardware

- Program control
- I/O
- File System
- Comms
- Error Mgmt
- Resource Mgmt
- Auditing
- Security
# ESnet’s Network Operating System

<table>
<thead>
<tr>
<th>OS Kernel</th>
<th>Net. OS equivalent</th>
<th>OS Kernel</th>
<th>Net. OS equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program control</td>
<td>Priority, Permissions,</td>
<td>Error Mgmt</td>
<td>perfSONAR, SNMP Traps,</td>
</tr>
<tr>
<td>I/O</td>
<td>Policy</td>
<td></td>
<td>Nagios</td>
</tr>
<tr>
<td>File System</td>
<td>MD-SAL and OpenFlow 1.3</td>
<td></td>
<td>QoS, Bandwidth/ Latency</td>
</tr>
<tr>
<td></td>
<td>(programs data flow)</td>
<td></td>
<td>OSCARS/NSI</td>
</tr>
<tr>
<td>Comms</td>
<td>Permanent store: Network</td>
<td></td>
<td>Logs at all levels</td>
</tr>
<tr>
<td></td>
<td>State, Configs, Policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mgmt. Network, SSH access, partitioning</td>
<td></td>
<td>Linux Security Model, ACLs</td>
</tr>
</tbody>
</table>

Sorted in alpha order by Net. OS equivalent.
What do the User Program’s interact with?

User APIs

System Calls

‘declarative’ interfaces or Intent-based Interfaces

List of composable network services for example: point to multi-point service
This is not an ‘Intent’ interface.

Now that you have logged on, click on the Start button and navigate to your email account. Take five minutes to read, then make a cup of tea...

To increase productivity, Rob’s boss decided to micro-manage his employees.
Intent versus Prescription

Intent

- What I want, not how to do it

- Portable, independent of protocol, vendor, media, etc.

- “I want my headache to stop”

- “Bob is allowed to access the internet”

- “Please cut my lawn”

Prescription

- How to do it (Commands, rules, settings)

- Non-portable, dependent on protocol, vendor, media, etc.

- “Give me two aspirin”

- “Send packets matching this {5-tuple} out port 11”

- “Take mower out of truck, fill gas and oil, pull starter cord, push onto lawn, ...”
How does it relate to the network?

- Multipoint VPN
  - ‘setup connectivity with isolation between points A, B and C on the topology’

- Traffic Tap
  - ‘Tap all TCP traffic going between Site A and Site B’

- Errors
  - ‘Alert me when packet loss exceeds 0.1%’

- Resiliency
  - ‘Create service backup, tolerant till 40% service degradation in bandwidth’
Constrained application-network interaction prevents benefits of automation, orchestration, optimization

High-level network abstractions for applications-network to have a ‘meaningful’ conversation

No abstractions for networks to feedback ‘service state’ to the application

Not possible as a network request today:
I would like to retrieve a PB of climate data by tomorrow morning from these five ESGF sites

“failed”
“congested”
“delayed”
Intent’s are next SDN frontier, lots of community interest and efforts

• **Congress** - Policy as a Service, uses SQL Policy Language, Interfaces to OpenStack modules

• **Group-based Policy (GBP)** - Implementations in Docker (preliminary) and in OpenStack (advanced).

• **Network Intent Composition (NIC)** - OpenDayLight project that can be implemented used independently and over GBP as well. *Boulder* in Open Source SDN / ONF

• **ONOS** – Carrier-grade SDN network operating system that enables intent via JAVA modules, accessed via REST APIs.

• **NEMO** - Network Modeling language which could serve as a Domain Specific Language (DSL) providing abstraction of network models and operation patterns. Uses JAVA, JS, YANG.

• **Frenetic** – Class Programmable-friendly domain specific language to build modular network applications providing powerful abstractions. Pyretic - implementation of frenetic in python
SDN Take-Away

• SDN is a journey R&E networks have been on for a while, but recently commercially formalized

• Innovator’s dilemma gaps between established and the new ways
  • industry and R&E momentum will close those gaps
  • Maturity will still take some time
  • Focus on the problem being solved

• Plan for the SDN future – skillsets, trying, training and hiring
Breaking News: L2 QoS enabled on ESnet’s SDN Testbed controlled with OF 1.3

- ESnet worked with Corsa to create a joint L2/L3 pipeline with L2 QoS controlled via OF 1.3 protocol

- Features enabled
  - Traffic isolation (guaranteed bandwidth circuits, best-effort circuits, best-effort IP, scavenger IP)
  - Prevent over-subscription of guaranteed bandwidth (using meters)

- After extensive testing, the SDN testbed has been upgraded with this feature
- Demonstration @ SC15
Thank you!

- For more details contact, imonga at es dot net