NETCONF and YANG integration in ONOS Southbound Interface

Andrea Campanella, ON.Lab
ONS 2016 -- ONOS Mini Summit
• Why NETCONF and YANG in ONOS
• ONOS Architecture
• Implementation goals and challenges
• NETCONF in ONOS
• Utilities for YANG
• Driver: tying everything together
• Accomplishments

• Looking Ahead
• Q&A
• Documentation
Why NETCONF?

• Broaden ONOS device configuration capabilities
  • enable high-level, protocol-agnostic interaction with devices

• Simple but powerful configuration and management protocol
  • RPC operations and device notifications
  • XML-defined messages
  • robust transactions for multiple devices

• Support multiple device families from different vendors
  • IETF standard
Why YANG?

- Device information and data model
  - configuration and state description
- High-level human readable language
- Modularity and extensibility
- Natural choice for NETCONF
  - one-to-one NETCONF XML mapping
- Widespread adoption
  - IETF defined data model language

YANG:
```yang
leaf host-name {
  type string;
  description "Hostname for this system";
}
```

XML:
```xml
<host-name>my.example.com</host-name>
```
ONOS Architecture

• **Scalability, High Availability & Performance**
  • sustain demands of service provider networks
• **Northbound & Southbound abstractions, modularity**
  • allow customization without changing the core code-base
• **Protocol and device model independency**
  • avoid protocol or model specifics and dependencies in the core
  • hidden complexity to upper layers
  • testability, extensibility, customization
NETCONF, YANG Implementation Goals

- Clean abstraction to upper layers.
- Maintain separation of concerns
  - avoid other types of communication
    -> use existing SB architecture
- Technology and protocol independency
  - no YANG/NETCONF in ONOS core.
- On-demand use and activation
- Core stays independent

This is where the specifics are contained
Implementation Challenges

• Translate YANG model to XML

• No standard content for NETCONF payload
  • the standard YANG IETF models are not used
  • always per-device models.
  • models overlap on features

• Lack of adequate tools for YANG
  • not general or standalone
  • not flexible enough

<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:NETCONF:base:1.0">
   ?
</rpc>
NETCONF Protocol library

• NETCONF protocol
  • self contained bundle
  • on demand activation
• Interaction via interfaces
• Modularity and extensibility
  • different transport protocol
  • different message handling
• NETCONF over SSH
  • secure and reliable
  • defined in RFC 6242
NETCONF SubController

- Java future with message-ids
  - Request-reply mechanism
  - enable both synchronous and asynchronous communication

- Listener mechanism for messages
  - device generated
  - notifications, alarms, shut-down

- Manages devices and connections
  - SSH session and connection
  - maintained state
  - periodical retry
New YANG utility implementation

- YANG to XML skeleton conversion
  - `onos-convert-YANG` bash script
  - decoupled from ONOS controller
- YANG XML Utility
  - `YangXmlUtils.java`
  - encoding-decoding facility
  - hides language complexity
- Stepstone for future YANG to JAVA generator
ONOS driver architecture outline

• Device specific driver
  • encapsulate specific logic and code
  • collection of behaviors
  • on-demand activation

• Abstraction via behaviors
  • define capabilities offered by the device
  • provide logic for operations
    • ports, controller, flowrule, power…

• Encapsulate interaction
  • protocol
  • information

<driver name="default" manufacturer="ON.Lab"
hwVersion="0.0.1" swVersion="0.0.1">
  <behaviour api=InterfacePath
    impl=ImplementationPath />
</driver>
NETCONF Drivers: tying everything together

• YANG, device information and NETCONF
• YANG Utils
  • YANG XML utilities
  • payload generation
• NETCONF
  • device specific calls
  • proper payload
• Abstraction of specific steps
  • operation results are returned
Accomplishments

• Use of NETCONF and YANG in drivers
  • well-defined interface
  • protocol and drivers on demand activation
  • extensibility

• Abstraction in Core maintained
  • no auto-generated code
  • no protocol or device specific logic

• Ease of use
  • device isolation
  • simple interaction with multiple and different devices
  • stepstone YANG tool for drivers
Final comments: looking ahead

• YANG to Java conversion
  • parser and translator

• Continue to make management plane easy and accessible
  • simple device and network configuration
  • abstract device access

• Define standard set of device commands
  • use standard IETF models
  • common YANG models
Q&A

ANY QUESTIONS?
• Get Started with ONOS
• ONOS NETCONF wiki
• ONOS YANG Southbound utilities wiki
• Future YANG to Java implementation
• ONOS architecture
• NETCONF RFC 6241
• NETCONF over SSH RFC 6242
• YANG RFC 6020