A High Level Rule-based Language for OpenFlow

Mehdi Mohammadi, Ph.D. Student
Mentors: Ala Al-Fuqaha, Ph.D.; James Yang, Ph.D.
Western Michigan University

Abstract

We studied the creation of rule-based Network policies to describe the controller's logic in Software Defined Network environments. We rely on existing OpenFlow controllers specifically Floodlight but the novelty of this project is to separate complicated language- and framework-specific APIs from policy descriptions. This separation makes it possible to extend the current work as a northbound higher level abstraction that can support a wide range of controllers that might potentially utilize different programming languages. This approach would enable network engineers to develop and deploy network control policies easier and faster.

OpenFlow High Level Languages

OpenFlow is a framework and protocol that realize the concept of Software-Defined Networking (SDN) in separating the data plane from the control plane. OpenFlow simplifies network management by providing high level abstractions to control a set of switches remotely.
In a typical OpenFlow network, if a switch can find a rule match to the received packet in its flow table, then it proceeds with that rule. Otherwise, the packet is sent to the controller for more processing. These rules are imposed by the programmer of the controller.
There are many OpenFlow controller frameworks that expose a high level abstraction above the OpenFlow APIs in their development frameworks such as POX, NOX, Beacon, Floodlight, Frenetic, Pyretic and Ryu.

Motivation and Significance of the Work

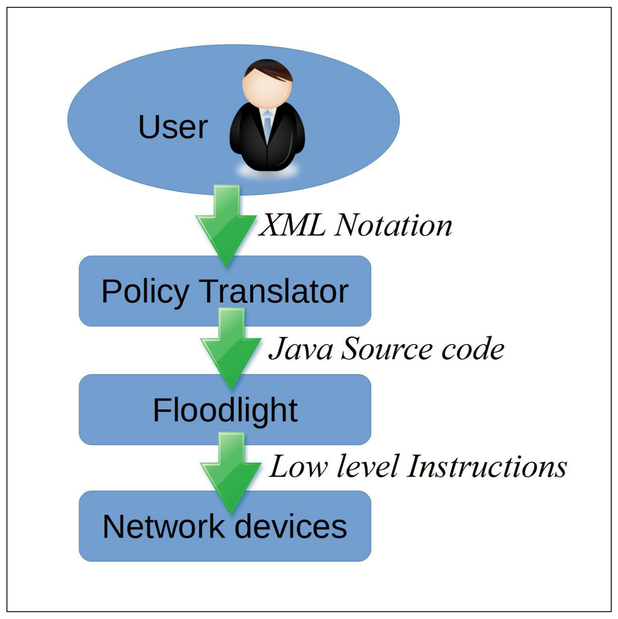
- ❑ An OpenFlow framework requires network engineers or administrators to write complete programs to control and manage data traffic in their network and control network devices.
- ❑ The programming languages that support OpenFlow are complex and administrators are required to know many irrelevant programming details.
- ❑ The problem is more prohibitive for a beginner network engineer who does not have a good background in the programming language of the controller.
- ❑ A simple and unified language in a higher abstraction level that does not expose all the bells and whistles of a complete programming language can fill this gap.
- ❑ In this work, we propose a rule-based syntax to fill that gap and it would serve as a user-friendly approach to describe the controller's policies.

Contribution of the Work

I. Introduced an XML-based scripting language for describing network control policies.
II. Implemented a translator that converts the XML file to Java source code containing the controller program for Floodlight.
III. This contributes to hide the complexities of the underlying APIs using a rule-based language.

Open Challenges

- Interaction between concurrent modules
- Support of low-level interface to switch hardware
- multi-tiered programming model



The overall architecture of the system

Language Specifications

- Rule description is defined in a hierarchical structure in which at the top level, we define the class name in the SDN element.
- Then a list of rules containing zero or more rule elements are declared.
- Inside each rule, one or more conditions are expressed. To have compositional conditions, condition elements support logical operators which are specified by the attribute "connector".
- The conditions themselves comply with a simple pattern "variable operator value".
- We designed a parser and code generator for that XML format using the YACC tool.

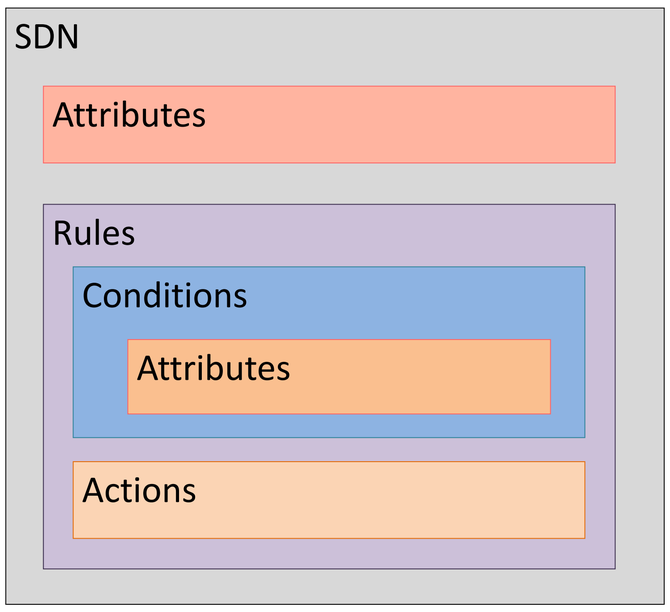
Experiment Design

- Generate the Java source code for a desired XML file
- Compile the generated Java files and deploy them on the controller machine.
- Configure this module to be loaded by the controller.
- We setup a simple network topology containing one switch (s0) and three hosts (h1, h2, h3) connected to the switch.
- Verify that the module is running (e.g., ping command).

Conclusions and Future Work

- ❑ A new approach to software-defined networking which presents a higher level of abstraction compared to current software-defined network programming languages was investigated.
- ❑ We defined a scripting language based on the XML notation that network administrators can utilize to define control policies without worrying about the complexities of the underlying controller framework. Indeed, this will make software-defined networking easier and more attractive for network administrators.
- ❑ This work opens up the opportunity of using service oriented architectures and web services as a model of collaboration between SDN controllers (e.g. controller offering load balancing or firewalling services).

Format of Policy Description file



Sample Application Scenarios

Express what you want:

- ✓ **All the packets that are destined to IP address 10.0.0.2 or are originated from IP address 192.168.0.1 should be forwarded to port 1 of the switch.**
- ✓ **Each packet originated from Telnet service (port 23) should be dropped.**

Describe it in XML:
The following two rules satisfy these tasks.

```
<SDN class="Demo">
  <rules>
    <rule>
      <condition>destIP=10.0.0.2
      </condition>
      <condition connector="or">
          srcIP=192.168.0.1
      </condition>
      <action>outPort=1</action>
    </rule>

    <rule>
      <condition>srcPort=23
      </condition>
      <action>outPort=0
      </action>
    </rule>
  </rules>
</SDN>
```

Comparison of XML and Java source code

	# Line of code	Size (KB)
XML file	18	1
Java file	147	5

GENI Engineering Conference 22
George Washington University – Washington, D.C,
March 23-26, 2015