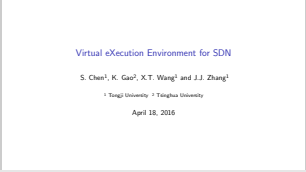
Virtual eXecution Environment for SDN

S. Chen[1], K. Gao[2], X.T. Wang[1] and J.J. Zhang[1]

[1] Tongji University [2] Tsinghua University

April 18, 2016

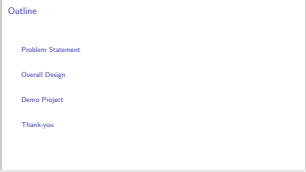
Virtual eXecution Environment for SDN



Hello, everyone! Nice to meet you all!

I'm Kai, a Ph.D student at Tsinghua University and my teammates are Shenshen, Tony and Jensen who are all from Tongji University. Today we are going to present the *Virtual eXecution Environment* for SDN programming, which is aimed to solve the data consistency problem in the control plane and to simplify programming in SDN.

Outline



Here is the outline of our presentation.
First we discuss a little bit about what kind of problems we are trying to solve and why current solutions are not good enough.
Second we introduce the overall design of our blueprint and show what we have implemented during the ONUG competition.
Finally we walk through the code and present a demo of the system.

Problem Statement

- Data consistency in the control plane
- Complexity in SDN programming

Problem Statement

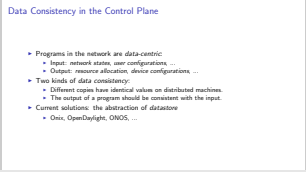
- Data consistency in the control plane
- Complexity in SDN programming

Our group has been working on the problem of SDN programming models, in the sense that we want to understand how SDN programming differs from generic programming by identifying domain-specific features, and to design a programming framework that satisfies the demand of correctness, efficiency and simplicity.

In this presentation we discuss mainly about two issues that are identified: the data consistency in the control plane and the programming complexity in modern SDN systems.

Data Consistency in the Control Plane

- Programs in the network are *data-centric*:
  - Input: *network states*, *user configurations*, ...
  - Output: *resource allocation*, *device configurations*, ...
- Two kinds of *data consistency*:
  - Different copies have identical values on distributed machines.
  - The output of a program should be consistent with the input.
- Current solutions: the abstraction of *datastore*
  - Onix, OpenDaylight, ONOS, ...



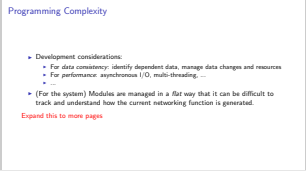
Most SDN applications depend on certain network states or user configurations and their final output includes resource allocations and forwarding rules on devices. All these input and output are modelled as *data* and the problem of data consistency has two meanings:
First, the data should be synchronized on different machines. Most modern controllers have provided mechanism such as distributed data base to solve this.
Second, the output of an application should be consistent with the input, which means if the input of the application changes, the output may need to be updated.

Modern SDN controllers have provided the abstraction of datastore to help solve these two consistency problems. However, the mechanisms they provide are low-level operations and make SDN programming complex.

Programming Complexity

- Development considerations:
  - For *data consistency*: identify dependent data, manage data changes and resources
  - For *performance*: asynchronous I/O, multi-threading, ...
  - ...
- (For the system) Modules are managed in a *flat* way that it can be difficult to track and understand how the current networking function is generated.

Expand this to more pages


Programming Complexity

- Development considerations:
  - For *data consistency*: identify dependent data, manage data changes and resources
  - For *performance*: asynchronous I/O, multi-threading, ...
  - ...
- (For the system) Modules are managed in a *flat* way that it can be difficult to track and understand how the current networking function is generated.

Expand this to more pages

The complexity of SDN programming comes from three aspects:
The development:
The debugging:
The system maintenance:
For example, to guarantee the data consistency we talked about in the last slide, programmers must identify dependent data in the program, register listeners to manage data changes and be very careful with allocated resources.

Motivations

- Dependent data can be identified *automatically*
-

Architecture

VXE in OpenDaylight

- The demo system is highly customized for OpenDaylight.

Components

Workflow

Q & A