Onix: A Distributed Control Platform for Large-scale Production Networks

Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, et al.
Operating Systems Design and Implementation (OSDI) 2010

2012. 04. 16

Myungchul Kwak
mckwak@mmlab.snu.ac.kr
Outline

- Introduction
- Onix design overview
- Evaluation
- Conclusion
Introduction

- Traditional computer networks lack a **general control paradigm**
  - Control plane mechanisms have advanced at a much slower pace
  - New control requirements have arisen
    - E.g. greater scale, increased security, VM migration

-> Growing movement towards new control paradigm
  - The control plane is **decoupled from the forwarding plane**
Software-Defined Networking

- A network-wide control platform is running on one or more servers in the network
  - Handles state distribution
  - Provides a programmatic interface
The SDN philosophy

- Basic primitives for state distribution should be implemented once in the **control platform**
  - Rather than separately for individual control tasks
- Should use well-known and general-purpose techniques from **distributed system literature**
  - Rather than more specialized algorithm

-> Allows to use a single control platform to implement a wide range of control functions
Control platform issues

- Generality
  - The control platform’s API must deliver a wide range of functionality

- Scalability
  - Any scaling limitations could be inherent problems of state management

- Control plane performance
  - Should not introduce additional control plane latencies

- Reliability

- Simplicity

-> Authors suggest the design of control platform satisfying all of these requirements - **Onix**
Outline

- Introduction
- Onix design overview
- Evaluation
- Conclusion
Four components of Onix

- Four components in a network controlled by Onix
  - Each have very distinct roles

![Diagram showing four components of Onix with network control logic, NIB, switch import/export, distribution I/E, management connectivity network infrastructure, and managed physical network infrastructure.]
The Onix API

- The principal contribution of Onix is defining a **useful** and **general API** for network control
  - Allowing control applications to read and write state to any element in the network

- This API consists of a **data model** that represents the network infrastructure
  - With each element corresponding to one or more **data objects**
  - Control logic can manage the current state with this data object

- Onix allows control logic to customize the data model
  - To support a wide range of control scenarios
The network state tracked by Onix is stored in a specific data structure – **the NIB**
- Roughly similar with RIB (Routing Information Base)

A Graph of all network entities within a network topology
- Network control applications are implemented by reading and writing to the NIB
- Onix provides **scalability** and **resilience** by replication and distributing the NIB
NIB details

- NIB holds a collection of network entities
  - Each holds a set of key-value pairs
  - Identified by a flat 128-bit global identifier
- Onix supports stronger typing through typed entities
  - Typed entities contain set of attributes and methods to perform operations

<table>
<thead>
<tr>
<th>Category</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>Find entities.</td>
</tr>
<tr>
<td>Create, destroy</td>
<td>Create and remove entities.</td>
</tr>
<tr>
<td>Access attributes</td>
<td>Inspect and modify entities.</td>
</tr>
<tr>
<td>Notifications</td>
<td>Receive updates about changes.</td>
</tr>
<tr>
<td>Synchronize</td>
<td>Wait for updates being exported to network elements and controllers.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Configure how state is imported to and exported from the NIB.</td>
</tr>
<tr>
<td>Pull</td>
<td>Ask for entities to be imported on-demand.</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Onix design overview
- Evaluation
- Conclusion
Evaluation

- Onix consists of roughly 150,000 lines of C++ and a number of third party libraries
  - Logic for communicating with the network elements
  - Aggregating this information into the NIB
  - A framework for application programmers
- Evaluates Onix through **Micro-benchmarks**
  - Designed to measure the performance of specific code
Throughput of the NIB

- The modifications of NIB involve only a short, fine-tuned code path
- Case of single attribute means throughput of threading library
Memory usage of Onix

- Onix can easily handle networks of millions of entities
Connection bandwidth

- Onix can perform well with 1000 connections roughly
  - It can forward over 100k packets per second
Conclusion

- This paper is not about the ideology of SDN, about its implementation
  - Authors present Onix as an existence proof that such control platforms are feasible
- Onix does not solve all the problems of network management
  - Need to learn how to build an efficient control logic, the management application