NOSIX

A Lightweight Portability Layer for the SDN OS

ONS 2013 - Research Track

Andreas Wundsam (Big Switch Networks) · Minlan Yu (USC) · Muruganantham Raju (USC)
Motivation

Core SDN promise:

Freedom from Vendor Lock-In

mix + match switches
reuse your SDN application
Motivation

Yet Unfulfilled!

Very difficult to write a truly portable* SDN application

(*) correct and efficient forwarding over a wide range of switches
Switch Diversity

Heterogenous Switch landscape!

**Data Plane:**
- Hardware vs. software
- # Flow Tables, Flow Table sizes
- Supported matches + actions

**Control Plane:**
- OpenFlow version + vendor extensions
- Rule updates (consistency, churn rate)
- Counters
Switch Diversity

Heterogenous Switch landscape!

**Data Plane:**
- Hardware vs. software
- # Flow Tables, Flow Table sizes
- Supported matches + actions

**Control Plane:**
- OpenFlow version + vendor extensions
- Rule updates (consistency, churn rate)
- Counters

Diversity is intrinsic:

Usage Scenarios, Price Points, Diversification
The Gap

Expectations of the application

switch feature-sets and performance characteristics
OpenFlow 1.0

• Enables **programmability**
• ‘Greatest Common Divisor’ feature set
  • 1 Flow table
  • Limited # actions
  • ....

✓ Portability (+- table size, L2 match...)

✗ Feature set
OpenFlow 1.1+

- Enables [way] more features
- Tons of options
  
  ✓ Feature set
  
  ✗ Portability
  
  ✗ Anybody implementing it?
OpenFlow 1.1+

- Enables more features
- Tons of options

Feature set ✓
Portability ❌
Anybody implementing it? ❌

OF still exposing “10% of switch brain?”
Higher Level Abstractions

• Examples
  • Onix: Network Graph (NIB)
  • Frenetic: Declarative Query Language
• Requires making assumptions
  → optimize for a particular programming model

✘ Generality, Expressiveness, Suitability for low-level tasks?
Trade-offs everywhere...

- Portability
- Abstraction Level
- Expressiveness
- Performance
- Predictability

Diagram showing the interrelation of these concepts.
NOSIX

a lightweight portability API in the controller
NOSIX
a lightweight portability API in the controller

Applications express expectations leverage application knowledge
NOSIX

a lightweight portability API
in the **controller**

Applications express expectations

leverage application knowledge

Drivers map to available features

leverage vendor knowledge

Switch Drivers

Application

NOSIX
Core Concepts: Top Down

Pipeline of VFTs
Virtualized Flow Tables

• Created by the Application
• Pipelined
• Default setting: ‘portability’
  • Full Feature Set
  • No resource constraints
• Annotations describe application expectations
Core Concepts: Top Down

VFT Annotations

- **Requirements**
  - **throughput** \( \geq 500 \text{ Mbit/s} \)
  - **churn** \( \geq 1000 \text{ flows/s} \)

- **Promises**
  - **only L2 matches** \( \leq 100 \text{ Flows/s} \)

- **Consistency**
Core Concepts: Bottom Up

Switch Drivers

- Map the annotated VFTs to the physical flow tables in the switch
- Use the annotations for optimized placement
Intuition

- Flows fall in natural groups
- Apps have information about the characteristics / allowable tradeoffs
Case Study:
Flow Table Size Limit in a Simulated P-Switch

Access Control → Microflows
80% small flows, 20% large flows
grow # flows > flow table size

NOSIX

Vft 1: Large

Vft 1: Small

frequent small flows high churn

high throughput low churn
Case Study: Flow Table Size Limit in a Simulated P-Switch

Access Control → Microflows
80% small flows, 20% large flows
grow # flows > flow table size

Baseline: Best effort

Vft 1: Large
Vft 1: Small

NOSIX

vs.
Case Study: Simulation Results

Dropped Bytes

Number of flows/sec

Dropped Bytes vs. Number of flows/sec

- NOSiX
- Baseline
Case Study: Simulation Results

The graph shows the number of flows dropped over the number of flows per second. Three scenarios are compared:

- **NOSIX: Small Flows**
- **NOSIX: Elephant Flows**
- **Baseline: Small Flows**
- **Baseline: Elephant Flows**

As the number of flows per second increases, the number of dropped flows also increases. The graph indicates that the NOSIX approach performs better in handling a large number of flows compared to the baseline, especially for small flows.
Case Study: Simulation Results

![Graph showing dropped flows over number of flows/sec]

- **Dropped flows**
- **Num of Flows dropped**
- **Number of flows/sec**

Legend:
- **NOSIX:Small Flows**
- **NOSIX:Elephant Flows**
- **Baseline:Small Flows**
- **Baseline:Elephant Flows**

**Large Flows:**
NOSIX keeps in Flow Table
Case Study: Simulation Results

- Small Flows: NOSIX forwards in Control Plane
- Large Flows: NOSIX keeps in Flow Table
Summary

• Lightweight portability API in the **controller**
• Applications express expectations
• Switch drivers implement them
• Addresses portability challenges in SDN
• Building block for higher abstraction level controllers