PISCES: A Programmable, Protocol-Independent Software Switch

P4 + OVS == Fast Forwarding!

Muhammad Shahbaz, Sean Choi, Ben Pfaff, Changhoon Kim, Nick Feamster, Nick McKeown, and Jennifer Rexford
Also appears at SIGCOMM 2016!

http://goo.gl/wmBmTu
Importance of Software Switches
Importance of Software Switches
Ease of Customization?

Enable **Rapid Development** and **Deployment** of Network Features!

Is it REALLY the case?
Ease of Customization?

For example, OVS supports following tunneling protocols:

- **VXLAN**: Virtual Extensible LAN
- **STT**: Stateless Transport Tunneling
- **NVGRE**: Network Virtualization Generic Routing Encapsulation

What about adding new protocols?
Rapid Development & Deployment?
Rapid Development & Deployment?

Requires domain expertise in:

- Network protocol design
- Software development
  - Develop
  - Test
  - Deploy
  ... large, complex codebases.

Arcane APIs
- Can take **3-6 months** to get a new feature in.
- **Maintaining changes** across releases
Rapid Development & Deployment?

Kernel

DPDK

Parser

Match-Action Pipeline
Rapid Development & Deployment?
Rapid Development & Deployment?

P4
- Parser
- Match-Action Pipeline

Compile

OVS
- Parser
- Match-Action Pipeline

DPDK

Native OVS

341 lines of code

14,535 lines of code
Rapid Development & Deployment?

Performance overhead!
What’s the cost of programmability on Performance?
PISCES: A Protocol-Independent Software Switch

PISCES
vSwitch

P4

OVS
PISCES: A Protocol-Independent Software Switch

P4

Compiler

parse match action

OVS

Runtime Flow Rules

Flow Rule Checker

Executable
PISCES: A Protocol-Independent Software Switch

- P4 and OVS packet forwarding models.

- Performance overhead of a naïve mapping from P4 to OVS.

- PISCES compiler optimizations to reduce the performance overhead.
P4 Forwarding Model (or Post-Pipeline Editing)

Ingress Packet

Packet Parser

Match-Action Tables

Packet Deparser

Egress Packet

Header Fields
OVS Forwarding Model (or Inline Editing)
(Modified) OVS Forwarding Model

- Supports both editing modes:
  - Inline Editing
  - Post-pipeline Editing
Naïve Mapping from P4 to OVS

A naïve compilation of L2L3-ACL benchmark application

Performance overhead of

~ 40%
Causes of Performance Degradation

Ingress → Packet Parser → Match-Action Pipeline → Packet Deparser → Egress

CPU Cycles per Packet
Causes of Performance Degradation

- Factors affecting CPU cycles:
  - Extra copy of headers in the post-pipeline editing mode
  - Fully-specified checksum calculation
  - Redundant parsing of header fields and more ...
Causes of Performance Degradation

Factor #1: **Extra copy of headers**

<table>
<thead>
<tr>
<th>Editing Mode</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post-Pipeline</td>
<td></td>
<td>Extra copy of headers</td>
</tr>
<tr>
<td>Inline</td>
<td><strong>No extra copy of headers</strong></td>
<td></td>
</tr>
</tbody>
</table>

- **Post-pipeline** editing consumes $2x$ more cycles than **inline** editing when parsing **VXLAN protocol**.
## Causes of Performance Degradation

### Factor #1: Extra copy of headers

<table>
<thead>
<tr>
<th>Editing Mode</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post-Pipeline</td>
<td>Packets are adjusted once</td>
<td>Extra copy of headers</td>
</tr>
<tr>
<td>Inline</td>
<td>No extra copy of headers</td>
<td>Multiple adjustments to packet</td>
</tr>
</tbody>
</table>

### Table

<table>
<thead>
<tr>
<th>Number of adjustments</th>
<th>Cycles per Packet (Post-Pipeline Editing)</th>
<th>Cycles per Packet (Inline Editing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deparse</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x1</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>x2</td>
<td>400</td>
<td>200</td>
</tr>
<tr>
<td>x4</td>
<td>600</td>
<td>300</td>
</tr>
<tr>
<td>x8</td>
<td>800</td>
<td>400</td>
</tr>
<tr>
<td>x16</td>
<td>1000</td>
<td>500</td>
</tr>
</tbody>
</table>

**Post-pipeline editing** vs **Inline editing**
Causes of Performance Degradation

Factor #2: **Fully-Specified Checksums**

[Diagram showing the process of packet parsing and parsing with an incremental checksum for TTL decrement.]
Causes of Performance Degradation

Factor #3: **Redundant parsing of headers**
## Optimizing for CPU Cycles

<table>
<thead>
<tr>
<th>Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline vs. post-pipeline editing</td>
</tr>
<tr>
<td>Incremental checksum</td>
</tr>
<tr>
<td>Parser specialization</td>
</tr>
<tr>
<td>Action specialization</td>
</tr>
<tr>
<td>Action coalescing</td>
</tr>
</tbody>
</table>
# Optimizing for CPU Cycles

<table>
<thead>
<tr>
<th>Optimizations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline vs. post-pipeline editing</td>
<td>Extra Copy of Headers</td>
</tr>
<tr>
<td>Incremental checksum</td>
<td>Fully-Specified Checksum</td>
</tr>
<tr>
<td>Parser specialization</td>
<td>Redundant Parsing</td>
</tr>
<tr>
<td>Action specialization</td>
<td></td>
</tr>
<tr>
<td>Action coalescing</td>
<td></td>
</tr>
</tbody>
</table>
Optimizing for CPU Cycles

Optimizations

- Inline vs. post-pipeline editing
- Incremental checksum
- Parser specialization
- Action specialization
- Action coalescing
Optimized Mapping from P4 to OVS

All optimizations together

Performance overhead of < 2%
Another Cause for Performance Degradation

Packet Deparser

Match-Action Cache

Match-Action Tables

Optimizations

Cached field modifications

Stage assignment

Ingress

Packet Parser

Cache Misses
Next Steps

- Support for **stateful memories** and **INT**

- **Integration** with the **mainline OVS**
  - Interning at VMware to make this happen!
Summary

With appropriate compiler optimizations ...

P4 + OVS == Fast Forwarding!
Questions?

Learn more and try PISCES here:

https://github.com/P4-vSwitch

Muhammad Shahbaz
mshahbaz@cs.princeton.edu