DevoFlow: Scaling Flow Management for High-Performance Networks

Andy Curtis
Jeff Mogul
Jean Tourrilhes
Praveen Yalagandula
Puneet Sharma
Sujata Banerjee
Software-defined networking
Software-defined networking

• Enables programmable networks
Software-defined networking

- Enables programmable networks
- Implemented by OpenFlow
Software-defined networking

- Enables programmable networks
- Implemented by OpenFlow
- OpenFlow is a great concept, but...
  - its original design imposes excessive overheads
Traditional switch

Control-plane

Data-plane
Traditional switch

Control-plane

Data-plane

Inbound packets

Routed packets

Wednesday, August 17, 11
Traditional switch

Reachability

Inbound packets

Control-plane

Data-plane

Reachability

Routed packets
Centralized controller

Flow setups
Link state
Forwarding rule stats
Forwarding table entries
Statistics requests

Inbound packets

Data-plane

Routed packets

Wednesday, August 17, 11
OpenFlow enables innovative management solutions
OpenFlow enables innovative management solutions

- Consistent routing and security policy enforcement [Ethane, SIGCOMM 2007]

- Data center network architectures like VL2 and PortLand [Tavakoli et al. Hotnets 2009]

- Client load-balancing with commodity switches [Aster*x, ACLD demo 2010; Wang et al., HotICE 2011]

- Flow scheduling [Hedera, NSDI 2010]

- Energy-proportional networking [ElasticTree, NSDI 2010]

- Automated data center QoS [Kim et al., INM/WREN 2010]
But OpenFlow is not perfect...

- Scaling these solutions to data center-sized networks is challenging
Contributions

• Characterize overheads of implementing OpenFlow in hardware

• Propose DevoFlow to enable cost-effective, scalable flow management

• Evaluate DevoFlow by applying it to data center flow scheduling
Contributions

- Characterize overheads of implementing OpenFlow in hardware

Experience drawn from implementing OpenFlow on HP ProCurve switches
OpenFlow couples flow setup with visibility.
If no forwarding table rule at switch (exact-match or wildcard)
Two problems arise...
**problem 1:**

bottleneck at controller
problem 1: bottleneck at controller

Up to 10 million new flows per second in data center with 100 edge switches [Benson et al. IMC 2010]
problem 1: bottleneck at controller

Up to 10 million new flows per second in data center with 100 edge switches [Benson et al. IMC 2010]

If controller can handle 30K flow setups/sec. then, we need at least 333 controllers!
Onix [Koponen et al. OSDI 2010]
HyperFlow [Tootoonchian and Ganjali, WREN 2010]
Devolved controller [Tam et al. WCC 2011]
problem 2:
stress on switch control-plane
Control-plane

Data-plane

Inbound packets → Routed packets

Wednesday, August 17, 11
Scaling problem: switches

- Inherent overheads
- Implementation-imposed overheads
Scaling problem: switches

- Inherent overheads
  - Bandwidth
    - OpenFlow creates too much control traffic
      ~1 control packet for every 2–3 data packets
  - Latency
- Implementation-imposed overheads
Scaling problem: switches

- Inherent overheads
- Implementation-imposed overheads
  - Flow setup
  - Statistics gathering
  - State size (see paper)
Flow setup

Client A  ProCurve 5406 zl switch  Client B

OpenFlow controller
Flow setup

ProCurve 5406 zl switch

We believe our measurement numbers are representative of the current generation of OpenFlow switches
Flow setup

Flow setups per sec.

5406 zl

275
Flow setup

We can expect up to 10K flow arrivals / sec.
[Benson et al. IMC 2010]

40x difference!
Flow setup

Flow setups per sec.

5406 zl

Expected

10,000

Too much latency: adds 2ms to flow setup
80 Mbps

Inbound packets

Data-plane

Switch CPU

Routed packets

Control-plane

Wednesday, August 17, 11
Stats-gathering

- Flow setups and stat-pulling compete for this bandwidth
Stats-gathering

- Flow setups and stat-pulling compete for this bandwidth
Stats-gathering

- Flow setups and stat-pulling compete for this bandwidth
  - 2.5 sec. to collect stats from the average data center edge switch
Can we solve the problem with more hardware?

- Faster CPU may help, but won’t be enough
  - Control-plane datapath needs at least two orders of magnitude more bandwidth

- Ethernet speeds accelerating faster than CPU speeds

- OpenFlow won’t drive chip-area budgets for several generations
Contributions

• Characterize overheads of implementing OpenFlow in hardware

• Propose DevoFlow to enable cost-effective, scalable flow management

• Evaluate DevoFlow by applying it to data center flow scheduling
Devolved OpenFlow

We devolve control over most flows back to the switches
DevoFlow design

• Keep flows in the data-plane
• Maintain just enough visibility for effective flow management
• Simplify the design and implementation of high-performance switches
DevoFlow mechanisms

- Control mechanisms
- Statistics-gathering mechanisms
DevoFlow mechanisms

- Control mechanisms
  - Rule cloning
    - ASIC clones a wildcard rule as an exact match rule for new microflows
DevoFlow mechanisms

- Rule cloning

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>src port</th>
<th>dst Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>129.100.1.5</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>src port</th>
<th>dst Port</th>
</tr>
</thead>
</table>

wildcard rules

exact-match rules
DevoFlow mechanisms

- Control mechanisms
  - Rule cloning

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>src port</th>
<th>dst Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>129.100.1.5</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>src port</th>
<th>dst Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DevoFlow mechanisms

- Control mechanisms
  - Rule cloning

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>src port</th>
<th>dst Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>129.100.1.5</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Wildcard rules

Exact-match rules
DevoFlow mechanisms

- Control mechanisms
  - Rule cloning

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>src port</th>
<th>dst Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>129.100.1.5</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>129.200.1.1</td>
<td>129.100.1.5</td>
<td>4832</td>
<td>80</td>
</tr>
</tbody>
</table>

exact-match rules

wildcard rules
DevoFlow mechanisms

- Control mechanisms
  - Rule cloning
    - ASIC clones a wildcard rule as an exact match rule for new microflows
  - Local actions
    - Rapid re-routing
      - Gives fallback paths for when a port fails
    - Multipath support
Control-mechanisms

- Rule cloning
  - ASIC clones a wildcard rule as an exact match rule for new microflows
- Local actions
  - Rapid re-routing
  - Gives fallback paths for when a port fails
  - Multipath support

1/3 1/6 1/2

Wednesday, August 17, 11
Statistics-gathering mechanisms
Statistics-gathering mechanisms

- Sampling
  - Packet header is sent to controller with 1/1000 probability
Statistics-gathering mechanisms

• Sampling
  – Packet header is sent to controller with 1/1000 probability

• Triggers and reports
  – Can set a threshold per rule; when threshold is reached, flow is setup at central controller
Statistics-gathering mechanisms

- **Sampling**
  - Packet header is sent to controller with 1/1000 probability

- **Triggers and reports**
  - Can set a threshold per rule; when threshold is reached, flow is setup at central controller

- **Approximate counters**
  - Tracks all flows matching a wildcard rule
Implementing DevoFlow

- Have not implemented in hardware
- Can reuse existing functional blocks for most mechanisms
Using DevoFlow

- Provides tools to scale your SDN application, but scaling is still a challenge
- Example: flow scheduling
  - Follows Hedera’s approach [Al-Fares et al. NSDI 2010]
Using DevoFlow: flow scheduling
Using DevoFlow: flow scheduling

• Switches use multipath forwarding rules for new flows
Using DevoFlow: flow scheduling

• Switches use multipath forwarding rules for new flows

• Central controller uses sampling or triggers to detect elephant flows
  - Elephant flows are dynamically scheduled by the central controller
  - Uses a bin packing algorithm, see paper
Evaluation

• How much can we reduce flow scheduling overheads while still achieving high performance?
Evaluation: methodology

• Custom built simulator
  - Flow-level model of network traffic
  - Models OpenFlow based on our measurements of the 5406 zl
Evaluation: methodology

- Custom built simulator
  - Flow-level model of network traffic
  - Models OpenFlow based on our measurements of the 5406 zl

17 Mbps
Evaluation: methodology

• Clos topology
  - 1600 servers
  - 640 Gbps bisection bandwidth
  - 20 servers per rack

• HyperX topology
  - 1620 servers
  - 405 Gbps bisection bandwidth
  - 20 servers per rack
Evaluation: methodology

• Workloads
  - Shuffle, 128 MB to all servers, five at a time
  - Reverse-engineered MSR workload [Kandula et al. IMC 2009]
Evaluation: methodology

- Workloads
  - Shuffle, 128 MB to all servers, five at a time
  - Reverse-engineered MSR workload [Kandula et al. IMC 2009]

- Based on two distributions: inter-arrival times and bytes per flow
Evaluation: methodology

- Schedulers
  - ECMP
  - OpenFlow
    - Coarse-grained using wildcard rules
    - Fine-grained using stat-pulling (i.e., Hedera [Al-Fares et al. NSDI 2010])
- DevoFlow
  - Statistics via sampling
  - Triggers and reports at a specified threshold of bytes transferred
Evaluation: metrics

- Performance
  - Aggregate throughput

- Overheads
  - Packets/sec. to central controller
  - Forwarding table size
Performance: Clos topology

Shuffle with 400 servers

 Aggregate Throughput (Gbps)

ECMP

Wildcard

OpenFlow-based

Stat-pulling

Sampling

DevoFlow-based

Thresholds

Wednesday, August 17, 11
Performance: Clos topology

Shuffle with 400 servers

37% increase

<table>
<thead>
<tr>
<th>Method</th>
<th>Aggregate Throughput (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP</td>
<td>200</td>
</tr>
<tr>
<td>Wildcard</td>
<td>150</td>
</tr>
<tr>
<td>Stat-pulling</td>
<td>230</td>
</tr>
<tr>
<td>Sampling</td>
<td>270</td>
</tr>
<tr>
<td>Thresholds</td>
<td>290</td>
</tr>
</tbody>
</table>

ECMP Wildcard, Stat-pulling: OpenFlow-based
Sampling, Thresholds: DevoFlow-based
Performance: Clos topology

MSR workload

<table>
<thead>
<tr>
<th>Method</th>
<th>Aggregate throughput (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP</td>
<td>200</td>
</tr>
<tr>
<td>Wildcard</td>
<td>190</td>
</tr>
<tr>
<td>Stat-pulling</td>
<td>180</td>
</tr>
<tr>
<td>Sampling</td>
<td>200</td>
</tr>
<tr>
<td>DevoFlow-based</td>
<td>200</td>
</tr>
</tbody>
</table>

Wednesday, August 17, 11
Performance: HyperX topology

Shuffle with 400 servers

Aggregate Throughput (Gbps)

- ECMP
- VLB
- Stat-pulling
- Sampling
- Thresholds

OpenFlow-based

DevoFlow-based
Performance: HyperX topology

Shuffle with 400 servers

55% increase

- ECMP
- VLB
- Stat-pulling
- Sampling
- Thresholds

Aggregate Throughput (Gbps)

OpenFlow-based

DevoFlow-based

Wednesday, August 17, 11
Performance: HyperX topology

Shuffle with 400 servers

55% increase

Aggregate Throughput (Gbps)

ECMP  VLB  Stat-pulling  Sampling  Thresholds

OpenFlow-based  DevoFlow-based

Wednesday, August 17, 11
Performance: HyperX topology

Shuffle with 400 servers

55% increase

Aggregate Throughput (Gbps)

ECMP

VLB

Stat-pulling

Sampling

Thresholds

OpenFlow-based

DevoFlow-based

Wednesday, August 17, 11
Performance: HyperX topology

Shuffle + MSR workload

Aggregate throughput (Gbps)

- ECMP
- VLB
- Stat-pulling
- Sampling
- Thresholds

OpenFlow-based
DevoFlow-based
Performance: HyperX topology

Shuffle + MSR workload

18% increase

Aggregate throughput (Gbps)

ECMP | VLB | Stat-pulling | Sampling | Thresholds
OpenFlow-based | DevoFlow-based

Wednesday, August 17, 11
Overheads: Control traffic

![Bar chart showing control traffic overheads for OpenFlow-based and DevoFlow-based systems.]

- **OpenFlow-based**
  - Wildcard: 483 packets/sec.
  - Stats-pulling: 7758 packets/sec.

- **DevoFlow-based**
  - Sampling: 705 packets/sec.
  - Threshold: 74 packets/sec.
Overheads: Control traffic

- **OpenFlow-based**
  - Wildcard: 483
  - Stats-pulling: 7758

- **DevoFlow-based**
  - Sampling: 705
  - Threshold: 74

No. packets/sec. to controller

70% reduction by using DevoFlow.

Wednesday, August 17, 11
Overheads: Control traffic

- Multipath wildcard rules and targeted stats collection reduces control traffic.

- OpenFlow-based:
  - Wildcard rules: 483 packets/sec.
  - Stats-pulling: 7758 packets/sec.

- DevoFlow-based:
  - Sampling: 705 packets/sec.
  - Threshold: 74 packets/sec.
Overheads: Flow table entries

<table>
<thead>
<tr>
<th>Method</th>
<th>Forwarding table entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wildcard</td>
<td>71</td>
</tr>
<tr>
<td>Stats-pulling</td>
<td>926</td>
</tr>
<tr>
<td>Sampling</td>
<td>8</td>
</tr>
<tr>
<td>Threshold</td>
<td>15</td>
</tr>
</tbody>
</table>

OpenFlow-based

DevoFlow-based

Wednesday, August 17, 11
Overheads: Flow table entries

- **Wildcard** (OpenFlow-based): 71 entries
- **Stats-pulling** (OpenFlow-based): 926 entries
- **Sampling** (DevoFlow-based): 8 entries
- **Threshold** (DevoFlow-based): 15 entries

70–150x reduction in table entries at average edge switch.
Evaluation: overheads

Control-plane bandwidth needed

Control plane throughput needed (Mbps)

Stat-pulling rate (s)

Percent of forwarding bandwidth needed for control plane bandwidth

95th percentile

99th percentile

Wednesday, August 17, 11
Evaluation: overheads

Control-plane bandwidth needed

- 95th percentile
- 99th percentile

Control plane throughput needed (Mbps)

Stat-pulling rate (s)

5406zI bandwidth

Percent of forwarding bandwidth needed for control plane bandwidth

Wednesday, August 17, 11
Evaluation: overheads

Control-plane bandwidth needed

This is to meet a 2ms switch-internal queuing deadline

Control plane throughput needed (Mbps)

Stat-pulling rate (s)

5406zI bandwidth
Conclusions

- OpenFlow imposes high overheads on switches
- Proposed DevoFlow to give tools to reduce reliance on the control-plane
- DevoFlow can reduce overheads by 10–50x for data center flow scheduling
Other uses of DevoFlow

• Client load-balancing (Similar to Wang et al. HotICE 2011)

• Network virtualization [Sherwood et al. OSDI 2010]

• Data center QoS

• Multicast

• Routing as a service [Chen et al. INFOCOM 2011]

• Energy-proportional routing [Heller et al. NSDI 2010]
Implementing DevoFlow

• Rule cloning:
  – May be difficult to implement on ASIC. Can definitely be done with use of switch CPU

• Multipath support:
  – Similar to LAG and ECMP

• Sampling:
  – Already implemented in most switches

• Triggers:
  – Similar to rate limiters
Flow table size

• Constrained resource
  - Commodity switches:
    32K–64K exact match entries
    ~1500 TCAM entries

• Virtualization may strain table size
  - 10s of VMs per machine implies >100K table entries
2.5 sec. to pull stats at average edge switch
Evaluation: methodology

- Workloads
  - Reverse-engineered MSR workload [Kandula et al. IMC 2009]
Evaluation: performance
Clos topology

Aggregate Throughput (Gbps)

- ECMP
- Wildcard 1s
- Pull-based 5s
- Sampling 1/1000
- Threshold 1MB

Workload

- shuffle, n=400
- MSR, 25% inter-rack
- MSR, 75% inter-rack
- MSR + shuffle, n = 400
- MSR + shuffle, n = 800

Wednesday, August 17, 11
Evaluation: overheads
Control traffic

- Wildcard
- Pull-based
- Sampling
- DevoFlow-based

No. packets/sec. to controller

<table>
<thead>
<tr>
<th>Time</th>
<th>MSR, 25% inter-rack</th>
<th>MSR, 75% inter-rack</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1s</td>
<td>504</td>
<td>29,451</td>
</tr>
<tr>
<td>1s</td>
<td>483</td>
<td>7,758</td>
</tr>
<tr>
<td>10s</td>
<td>446</td>
<td>4,871</td>
</tr>
<tr>
<td>0.1s</td>
<td>1</td>
<td>7,123</td>
</tr>
<tr>
<td>1/100</td>
<td>1/1000</td>
<td>709</td>
</tr>
<tr>
<td>1/1000</td>
<td>1/10000</td>
<td>71</td>
</tr>
<tr>
<td>128KB</td>
<td>432</td>
<td>181</td>
</tr>
<tr>
<td>1MB</td>
<td></td>
<td>74</td>
</tr>
</tbody>
</table>

Wildcard, Pull-based, and Sampling are compared with Threshold. DevoFlow-based shows lower overheads compared to OpenFlow-based.
Evaluation: overheads
Flow table entries

DevoFlow aggressively uses multipath wildcard rules