Software-defined Networking and OpenFlow

POMI Workshop, April 2009

Nick McKeown
nickm@stanford.edu

OpenFlow team: Nick McKeown, Guido Appenzeller, Guru Parulkar, Brandon Heller, Glen Gibb, Masayoshi Kobayashi, Tatsuya Yabe, Mikio Hara, Rob Sherwood, Srini Seetharaman, David Underhill, Dave Erickson.
Part 1: Software-defined networking

• Trend towards defining the network infrastructure in software
• Requires a simple hardware substrate

Part 2: OpenFlow as an example
OS abstracts hardware substrate
→ Innovation in applications
Simple, common, stable, hardware substrate below
+ Programmability
+ Competition
→ Innovation in OS and applications
Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Innovation in infrastructure
A simple stable common substrate

1. **Allows applications to flourish**
   Internet: Stable IPv4 lead to the web

2. **Allows the infrastructure on top to be defined in software**
   Internet: Routing protocols, management, …

3. **Rapid innovation of the infrastructure itself**
   Internet: ...? What’s missing?
Mid-1990s:
“To enable innovation in the network, we need to program on top of a simple hardware datapath”

Active networking

Problems: isolation, performance, complexity
Late-1990s:
“To enable innovation in the network, we need the datapath substrate to be programmable”

Network processors

Problem: Accelerated complexity of the datapath substrate
Where we are

Many complex functions baked into the infrastructure
OSPF, BGP, multicast, differentiated services,
Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …

Very hard to change
Substrate is getting more and more complex
In networking, despite several attempts…

We’ve never agreed upon a clean separation between:

1. A simple common hardware substrate
2. And an open programming environment on top

But there are rumblings in large data centers, and service provider networks.
Observations

Prior attempts have generally

1. Assumed the current IP substrate is fixed, and tried to program it externally
   - But the substrate now consists of Ethernet, TCP, …

2. Defined the programming and control model up-front
   - But to pick the right x86 instruction set, Intel didn’t define Windows XP, Linux or VMware
We need…

- A simple hardware substrate that generalizes, subsumes and simplifies the current substrate
- A clean separation between the substrate and an open programming environment
- Very few preconceived ideas about how the substrate will be programmed
- Strong isolation
Substrate today

<table>
<thead>
<tr>
<th>Ethernet</th>
<th>IP</th>
<th>TCP</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA, SA, etc</td>
<td>DA, SA, etc</td>
<td>DP, SP, etc</td>
<td></td>
</tr>
</tbody>
</table>

Collection of bits to plumb flows (of different granularities) between end points
What is a flow?
- Application flow
- All http
- Peter’s traffic
- All packets to Canada
- ...

Types of action
- Allow/deny flow
- Route & re-route flow
- Isolate flow
- Make flow private
- Remove flow

We need flexible definitions of a flow

We don’t need many types of action
1. Unicast

2. Multicast
3.
- Multipath
  - Load-balancing
  - Redundancy

4.
- Waypoints
  - Middleware
  - Intrusion detection
  - …
Substrate: “Flowspace”

Collection of bits to plumb flows (of different granularities) between end points
Flows: Simple example

- Single flow
- All flows from A
- All flows between two subnets
Flows: Generalization

Field 1

Field n

Field 2

Single flow

Set of flows
Properties of Flowspace

**Backwards compatible**
- Current layers are a special case

**Easily implemented in hardware**
- e.g. TCAM flow-table in each switch

**Strong isolation of flows**
- Simple geometric construction
- Can prove which flows can/cannot communicate
A substrate

Flow-based

Small number of actions for each flow

- Plumbing: Forward to port(s)
- Control: Forward to controller
- Routing between flow-spaces: Rewrite header
- Bandwidth isolation: Min/max rate

External open API to flow-table
Part 1: Software-defined networking
• Trend towards defining the infrastructure in software
• Requires a simple hardware substrate

Part 2: OpenFlow as an example
Step 1:
Separate intelligence from datapath

Operators, users, 3rd party developers, researchers, …
Step 2: Cache decisions in minimal datapath

“If header = x, send to port 4”
“If header = y, overwrite header with z, send to ports 5,6”
“If header = ?, send to me”
Our Approach

1. Define the substrate
   ✓ Define the OpenFlow feature
   ✓ **First version (now):** OpenFlow-enabled switches
     Make it easy to add to commercial switches, routers, APs and basestations
   ✓ **Second version (~2yrs):** OpenFlow-optimized switches in general “flowspace”

2. Deploy on college campuses
3. Deploy in national backbone networks
4. Enable researchers to freely innovate on top
OpenFlow Basics
Ethernet Switch
Control Path (Software)

Data Path (Hardware)
Exploit the flow table in switches, routers, and chipsets

<table>
<thead>
<tr>
<th>Flow 1.</th>
<th>Rule (exact &amp; wildcard)</th>
<th>Action</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 2.</td>
<td>Rule (exact &amp; wildcard)</td>
<td>Action</td>
<td>Statistics</td>
</tr>
<tr>
<td>Flow N.</td>
<td>Rule (exact &amp; wildcard)</td>
<td>Default Action</td>
<td>Statistics</td>
</tr>
</tbody>
</table>
## Flow Table Entry

**OpenFlow Protocol**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Action</th>
<th>Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Packet + byte counters</td>
</tr>
</tbody>
</table>

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ mask what fields to match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Examples

### Switching

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td>00:1f..</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6</td>
</tr>
</tbody>
</table>

### Flow Switching

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>port3</td>
<td>00:2e..</td>
<td>00:1f..</td>
<td>0800</td>
<td>vlan1</td>
<td>1.2.3.4</td>
<td>5.6.7.8</td>
<td>4</td>
<td>17264</td>
<td>80</td>
<td>port6</td>
</tr>
</tbody>
</table>

### Firewall

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>22</td>
</tr>
</tbody>
</table>
## Examples

### Routing

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6</td>
</tr>
</tbody>
</table>

### VLAN

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6, port7, port9</td>
</tr>
</tbody>
</table>
Enable Innovation in your Network.

OpenFlow is an open standard that allows you to run experimental protocols in production networks. It is in the process of being implemented by major switch vendors and used today by universities to deploy innovative networking technology in their campus networks.
OpenFlow Usage

Dedicated OpenFlow Network

Controller

Peter’s code

OpenFlow Protocol

PC

OpenFlow Switch

Rule

Action

Statistics

Rule

Action

Statistics

Rule

Action

Statistics

Peter

Peter's code

OpenFlow Protocol

PC
Usage examples

Peter’s code:

► Static “VLANs”
► His own new routing protocol: unicast, multicast, multipath, load-balancing
► Network access control
► Home network manager
► Mobility manager
► Energy manager
► Packet processor (in controller)
► IPvPeter
► Network measurement and visualization
► …
Virtualizing OpenFlow

- Aaron’s Controller
- Heidi’s Controller
- Craig’s Controller

OpenFlow Switch

OpenFlow Protocol

OpenFlow FlowVisor & Policy Control

OpenFlow Switch

OpenFlow Protocol

OpenFlow Switch
Virtualizing OpenFlow

- OpenFlow Switch
- Broadcast
- Multicast
- http Load-balancer
- OpenFlow Protocol

FlowVisor & Policy Control

OpenFlow Switch

Virtualizing OpenFlow

Load-balancer

OpenFlow Protocol

OpenFlow Protocol
Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Faster innovation
OpenFlow Status
OpenFlow Hardware

Juniper MX

NEC IP8800

WiMax (NEC)

HP Procurve 5400

Cisco Catalyst 6k

PC Engines

Quanta LB4

More coming soon...
Stanford Deployment

Phase 1 (ongoing)
- Gates Building, 3A Wing only
  - Two switches (HP ProCurve 5400)
- ~30 Wireless APs
- ~25 users

Phase 2 (1H2009)
- Gates Building, All Floors
- 23 Switches (HP ProCurve 5400)
- Wireless TBD
- Hundreds of users

Phase 3 (2H2009)
- Packard and CIS Buildings
- Switch Count TBD (HP ProCurve 5400)
- Wireless TBD
- > 1000 users
Two Larger OpenFlow Deployments

Campus Trials
- At seven leading campuses with researchers and CIO
- Potentially 20-30 campuses to follow
Open up campus networks for innovations
Build robust OpenFlow infrastructure

Nation-wide OpenFlow substrate – connect campuses
- Internet2 backbone and six regional networks
- Clean slate inter-domain system
- Unified control of packet and circuit networks

Potentially funded by NSF/GENI in partnership with campuses, vendors, Internet2/NLR and regionals
Thank You!