Outline

• SDN Basics
  – Concepts
  – OpenFlow
  – Controller: Floodlight
  – OF-Config
  – Mininet

SDN Concepts

• What is software defined networking?
• Why SDN?
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation
Huge industry

Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

Source: Nick McKeown, Stanford
Million of lines of source code
6,000 RFCs
Billions of gates
Bloated
Power Hungry

• Vertically integrated, complex, closed, proprietary
• Networking industry with “mainframe” mind-set

The network is changing

Source: Nick McKeown, Stanford
Software Defined Network (SDN)

1. Open interface to packet forwarding
2. At least one Network OS probably many. Open- and closed-source
3. Consistent, up-to-date global network view

Network OS

**Network OS**: distributed system that creates a consistent, up-to-date network view
- Runs on servers (controllers) in the network
- Floodlight, POX, Pyretic, Nettle ONIX, Beacon, ... + more

Uses forwarding abstraction to:
- Get state information **from** forwarding elements
- Give control directives **to** forwarding elements

Source: Nick McKeown, Stanford
Software Defined Network (SDN)

Control Program

Control program operates on view of network
- **Input**: global network view (graph/database)
- **Output**: configuration of each network device

Control program is not a distributed system
- Abstraction hides details of distributed state
Forwarding Abstraction

Purpose: Abstract away forwarding hardware
Flexible
  – Behavior specified by control plane
  – Built from basic set of forwarding primitives
Minimal
  – Streamlined for speed and low-power
  – Control program not vendor-specific

OpenFlow is an example of such an abstraction

Why SDN?
Great talk by Scott Shenker
http://www.youtube.com/watch?v=WVs7Pc99S7w
(Story summarized here)
Networking

Networking is “Intellectually Weak”
Networking is behind other fields
Networking is about the mastery of complexity

Good abstractions tame complexity
Interfaces are instances of those abstractions

No abstraction => increasing complexity
We are now at the complexity limit

By comparison: Programming

Machine languages: no abstractions
   – Had to deal with low-level details
Higher-level languages: OS and other abstractions
   – File system, virtual memory, abstract data types, ...
Modern languages: even more abstractions
   – Object orientation, garbage collection,...
Programming Analogy

What if programmers had to:
- Specify where each bit was stored
- Explicitly deal with internal communication errors
- Within a programming language with limited expressability

Programmers would redefine problem by:
- Defining higher level abstractions for memory
- Building on reliable communication primitives
- Using a more general language

Specification Abstraction

Network OS eases implementation
Next step is to ease specification

Provide abstract view of network map
Control program operates on abstract view
Develop means to simplify specification
Software Defined Network (SDN)

Outline

• SDN Basics
  – Concepts
  – OpenFlow
  – Switches and Controllers
  – OF-Config
  – Mininet

Source: Nick Mckeown, Stanford
OpenFlow

- Why OpenFlow?
- How does OpenFlow work?

Why OpenFlow?
The Ossified Network

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

- An industry with a “mainframe-mentality”, reluctant to change

---

Research Stagnation

- Lots of *deployed* innovation in other areas
  - OS: filesystems, schedulers, virtualization
  - DS: DHTs, CDNs, MapReduce
  - Compilers: JITs, vectorization
- Networks are largely the same as years ago
  - Ethernet, IP, WiFi
- Rate of change of the network seems slower in comparison
  - Need better tools and abstractions to demonstrate and deploy
Closed Systems (Vendor Hardware)

• Stuck with interfaces (CLI, SNMP, etc)
• Hard to meaningfully collaborate
• Vendors starting to open up, but not usefully
• Need a fully open system – a Linux equivalent

Open Systems

<table>
<thead>
<tr>
<th></th>
<th>Performance Fidelity</th>
<th>Scale</th>
<th>Real User Traffic?</th>
<th>Complexity</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>medium</td>
<td>medium</td>
<td>no</td>
<td>medium</td>
<td>yes</td>
</tr>
<tr>
<td>Emulation</td>
<td>medium</td>
<td>low</td>
<td>no</td>
<td>medium</td>
<td>yes</td>
</tr>
<tr>
<td>Software Switches</td>
<td>poor</td>
<td>low</td>
<td>yes</td>
<td>medium</td>
<td>yes</td>
</tr>
<tr>
<td>NetFPGA</td>
<td>high</td>
<td>low</td>
<td>yes</td>
<td>high</td>
<td>yes</td>
</tr>
<tr>
<td>Network Processors</td>
<td>high</td>
<td>medium</td>
<td>yes</td>
<td>high</td>
<td>yes</td>
</tr>
<tr>
<td>Vendor Switches</td>
<td>high</td>
<td>high</td>
<td>yes</td>
<td>low</td>
<td>no</td>
</tr>
</tbody>
</table>

Source: Big Switch Networks

gap in the tool space
none have all the desired attributes!
Ethane, a precursor to OpenFlow

Centralized, reactive, per-flow control

OpenFlow: a pragmatic compromise

- + Speed, scale, fidelity of vendor hardware
- + Flexibility and control of software and simulation
- Vendors don’t need to expose implementation
- Leverages hardware inside most switches today (ACL tables)

See Ethane SIGCOMM 2007 paper for details
How does OpenFlow work?

https://www.opennetworking.org
OpenFlow Example

Flow Table

<table>
<thead>
<tr>
<th>MAC src</th>
<th>MAC dst</th>
<th>IP Src</th>
<th>IP Dst</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>5.6.7.8</td>
<td></td>
<td>port 1</td>
</tr>
</tbody>
</table>

Software Layer

Hardware Layer

Controller

OpenFlow Basics

Flow Table Entries

Rule | Action | Stats
---|---|---

Packet + byte counters

1. Forward packet to zero or more ports
2. Encapsulate and forward to controller
3. Send to normal processing pipeline
4. Modify Fields
5. Any extensions you add!

Switch Port | VLAN ID | VLAN pcp | MAC src | MAC dst | Eth type | IP Src | IP Dst | IP ToS | IP Prot | L4 sport | L4 dport
---|---|---|---|---|---|---|---|---|---|---|---

+ mask what fields to match
### 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>00:1f:..</td>
<td>*</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>00:20:.. 00:1f: 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>
<td></td>
<td></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>22</td>
<td>drop</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>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6</td>
<td></td>
</tr>
</tbody>
</table>

#### VLAN 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>00:1f:..</td>
<td>vlan1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6, port7, port9</td>
<td></td>
</tr>
</tbody>
</table>
Centralized vs Distributed Control
Both models are possible with OpenFlow

Centralized Control

Distributed Control

Flow Routing vs. Aggregation
Both models are possible with OpenFlow

Flow-Based
- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

Aggregated
- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone
Reactive vs. Proactive (pre-populated)
Both models are possible with OpenFlow

Reactive
- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

Proactive
- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules

Usage examples
- Alice’s code:
  - Simple learning switch
  - Per Flow switching
  - Network access control/firewall
  - Static “VLANs”
  - Her own new routing protocol: unicast, multicast, multipath
  - Home network manager
  - Packet processor (in controller)
  - IPvAlice

- VM migration
- Server Load balancing
- Mobility manager
- Power management
- Network monitoring and visualization
- Network debugging
- Network slicing

... and much more you can create!
What can you not do with OpenFlow ver1.0

- Non-flow-based (per-packet) networking
  - ex. Per-packet next-hop selection (in wireless mesh)
  - yes, this is a fundamental limitation
  - BUT OpenFlow can provide the plumbing to connect these systems
- Use all tables on switch chips
  - yes, a major limitation (cross-product issue)
  - BUT OpenFlow 1.3 version will expose these

What can you not do with OpenFlow ver1.0

- New forwarding primitives
  - BUT provides a nice way to integrate them through extensions
- New packet formats/field definitions
  - BUT a generalized OpenFlow (2.0) is on the horizon
- Optical Circuits
  - BUT efforts underway to apply OpenFlow model to circuits
- Low-setup-time individual flows
  - BUT can push down flows proactively to avoid delays
Where it’s going

• OF v1.3: Spring 2013
  – multiple tables: leverage additional tables
  – tags and tunnels
  – multipath forwarding
  – per flow meters
• OF v2+
  – generalized matching and actions: protocol independent forwarding

Outline

• SDN Basics
  – Concepts
  – OpenFlow
    – Switches and Controllers
  – OF-Config
  – Mininet
Switches and Controllers

- OpenFlow switches and vendors
- Controllers
  - Floodlight
## Current SDN hardware

<table>
<thead>
<tr>
<th>Model</th>
<th>Virtualize</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juniper MX-series</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC IP8000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIMax (NEC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP Procurve 5400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netgear 7324</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC Engines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pronto 3240/3290</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ciena Coredirector</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More coming soon...

## Commercial Switch Vendors

<table>
<thead>
<tr>
<th>Model</th>
<th>Virtualize</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP Procurve 5400z1 or 6600</td>
<td>1 OF instance per VLAN</td>
<td>-LACP, VLAN and STP processing before OpenFlow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Wildcard rules or non-IP pkts processed in s/w</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Header rewriting in s/w</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-CPU protects mgmt during loop</td>
</tr>
<tr>
<td>NEC IP8800</td>
<td>1 OF instance per VLAN</td>
<td>-OpenFlow takes precedence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Most actions processed in hardware</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-MAC header rewriting in h/w</td>
</tr>
<tr>
<td>Pronto 3240 or 3290 with Pica8 or Indigo firmware</td>
<td>1 OF instance per switch</td>
<td>-No legacy protocols (like VLAN and STP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Most actions processed in hardware</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-MAC header rewriting in h/w</td>
</tr>
</tbody>
</table>
Controller Vendors

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Notes</th>
</tr>
</thead>
</table>
| Nicira’s NOX      | • Open-source GPL  
|                   | • C++ and Python  
|                   | • Researcher friendly                                                |
| Nicira’s ONIX     | • Closed-source  
|                   | • Datacenter networks                                                |
| SNAC              | • Open-source GPL  
|                   | • Code based on NOX0.4  
|                   | • Enterprise network  
|                   | • C++, Python and Javascript  
|                   | • Currently used by campuses                                        |
| Stanford’s Beacon | • Open-source  
|                   | • Researcher friendly  
|                   | • Java-based                                                        |
| BigSwitch controller | • Ha open source version  
|                   | • Based on Beacon  
|                   | • Enterprise network                                                |
| Maestro (from Rice Univ) | • Open-source  
|                   | • Based on Java                                                     |
| Frenetic or Nettle | • Open-source  
|                   | • Written in functional programming languages                       |

Floodlight Architecture

Overview

- Floodlight is a collection of modules
- Some modules (not all) export services
- All modules in Java
- Rich, extensible REST API
**Floodlight Architecture**

**Module descriptions**

- **FloodlightProvider** *(IFloodlightProviderService)*
  - Translates OF messages to Floodlight events
  - Managing connections to switches via Netty

- **TopologyManager** *(ITopologyManagerService)*
  - Computes shortest path using Dijkstra
  - Keeps switch to cluster mappings

- **LinkDiscovery** *(ILinkDiscoveryService)*
  - Maintains state of links in network
  - Sends out LLDPs

- **Forwarding**
  - Installs flow mods for end-to-end routing
  - Handles island routing

- **DeviceManager** *(IDeviceService)*
  - Tracks hosts on the network
  - MAC -> switch,port, MAC->IP, IP->MAC

- **StorageSource** *(IStorageSourceService)*
  - DB style storage (queries, etc)
  - Modules can access all data and subscribe to changes

- **RestServer** *(IRestApiService)*
  - Implements via Restlets (restlet.org)
  - Modules export RestletRoutable

- **StaticFlowPusher** *(IStaticFlowPusherService)*
  - Supports the insertion and removal of static flows
  - REST-based API

- **VirtualNetworkFilter** *(IVirtualNetworkFilterService)*
  - Create layer 2 domain defined by MAC address
  - Used for OpenStack / Quantum

---

**Floodlight Programming Model**

**Northbound APIs**

- IFloodlightModule
  - Java module that runs as part of Floodlight
  - Consumes services and events exported by other modules
    - OpenFlow (ie, Packet-in)
    - Switch add / remove
    - Device add /remove / move
    - Link discovery

**External Application**

- Communicates with Floodlight via REST
  - Quantum / Virtual networks
  - Normalized network state
  - Static flows
REST API Reference

A moving target...but...

<table>
<thead>
<tr>
<th>Network State</th>
<th>Static Flows</th>
<th>Virtual Network</th>
<th>User Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Hosts</td>
<td>Add Flow</td>
<td>Create Network</td>
<td></td>
</tr>
<tr>
<td>List Links</td>
<td>Delete Flow</td>
<td>Delete Network</td>
<td></td>
</tr>
<tr>
<td>List Switches</td>
<td>List Flows</td>
<td>Add Host</td>
<td></td>
</tr>
<tr>
<td>GetStats (DPID)</td>
<td>RemoveAll Flows</td>
<td>Remove Host</td>
<td></td>
</tr>
<tr>
<td>GetCounters (OFType...)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Floodlight Controller

Switch

vSwitch

Switch

Programming Floodlight

Using the REST API

- Fine-grained ability to push flows over REST
- Access to normalized topology and device state
- Extensible access to add new APIs

```
importurllib
importjson

class StaticFlowPusher(object):
    def __init__(self, server):
        self.server = server
    def set(self, data):
        path = '/api/statisticflowentry/push

```
Programming Floodlight

Creating a module

- Handle OpenFlow messages directly (ie. PacketIn)
- Expose services to other modules
- Add new REST APIs

Outline

- SDN Basics
  - Concepts
  - OpenFlow
  - Switches and Controllers
  - OF-Config
  - Mininet
OpenFlow configuration and Management Protocol

- Bootstrap OpenFlow network
  - Switch connects to controller
  - Controller(s) to connect to must be configured at switches

- Allocate resources within switches
  - Ports
  - Queues
  - . . .

OpenFlow configuration and Management Protocol: Reference Model

- Configuration Point
  - Source of switch configuration
  - OpenFlow Capable Switch
    - Hosts one or more logical switches

- OpenFlow Controller
  - OpenFlow Logical Switch
    - Instance of an OpenFlow Switch

- OpenFlow Capable Switch
  - Resources (ports, queues)
  - Using IETF Netconf & XML data models
OF-CONFIG Scope and Releases

- **OF-CONFIG 1.0** (Jan 2012) based on OpenFlow 1.2
  - assigning controllers to logical switches
  - retrieving assignment of resources to logical switches
  - configuring some properties of ports and queues

- **OF-CONFIG 1.1** (Apr 2012) based on OpenFlow 1.3
  - added controller certificates and resource type "table"
  - retrieving logical switch capabilities signaled to controller
  - configuring of tunnel endpoints

- **OF-CONFIG 1.1.1** (Aug 2012) based on OpenFlow 1.3.1
  - consolidation of version 1.1, fixing small inconsistencies

- **OF-CONFIG 1.2** (early 2013) based on OpenFlow 1.3.1
  - features still under discussion, candidates include
    - retrieving capable switch capabilities, configuring logical switch capab.
    - assigning resources to logical switches
    - simple topology detection
    - event notification

Use of Netconf and Yang

- Netconf was chosen as management protocol
  - not necessarily accepted as ideal solution
  - still discussing alternatives

- XML schema was chosen as modeling language
  - Yang is also used, but XML is normative
  - normative XML schema generated from Yang code

- So far, the focus has been on configuration
  - bootstrap of an OpenFlow network is the obvious first thing to do

- New work items will be more on OAM
  - incl. event notifications
Outline

• SDN Basics
  – Concepts
  – OpenFlow
  – Switches and Controllers
  – OF-Config
  – Mininet

Mininet

• Machine-local virtual network
  – great dev/testing tool
• Uses linux virtual network features
  – Cheaper than VMs
• Arbitrary topologies, nodes
Mininet (Cont’d)

• Rapidly prototype, develop and test
  – Interestingly-sized networks (16-100 nodes) start up in seconds
  – No lengthy lab reconfiguration or rebooting required
  – Always-accessible network resources, in any topology, at essentially no cost
  – Designs that work on Mininet transfer seamlessly to hardware for full speed operation

Mininet (Cont’d)

• Repeatably test, analyze, and predict network behavior
  – Easy replication of experimental and test results
  – Examine effects of code or network changes before testing/deploying on hardware
  – Allows automated system-level tests and experiments
  – Recreate real-world network and test cases for a variety of topologies and configurations
Mininet (Cont’d)

• Quickly get up and running
  – Free and permissively licensed (BSD)
  – Minimal hardware requirements
  – Accessible to novices thanks to simple CLI
  – Smooth learning curve thanks to walkthrough, tutorial, examples and API documentation
  – Strong users and support community

Questions?