Software-Defined Networking (SDN): Background
Introduction

- State of the Internet
- The Need for a New Network
- Software-Defined Networking (SDN)
- Network Data Plane
- SDN Data Plane Technology: OpenFlow
STATE OF THE INTERNET
Network OSI Model

• Proposed in the late 1970s
• Open Systems Interconnection model:
  – It characterizes and standardizes a communication system by dividing it into layers
  – A layer serves the layer above it and is served by the layer below it
• Two major components:
  – An abstract model of networking
  – A set of protocols
## Network OSI Model

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>Data Unit</th>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Layers</td>
<td>Data</td>
<td>7. Application</td>
<td>Network process to application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Presentation</td>
<td>Data presentation, encryption and decryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. Session</td>
<td>Interhost communication</td>
</tr>
<tr>
<td>Segment</td>
<td>Packet/Datagram</td>
<td>3. Network</td>
<td>Addressing, routing and delivery of datagrams</td>
</tr>
<tr>
<td>Media Layers</td>
<td>Bit/Frame</td>
<td>2. Data Link</td>
<td>Reliable direct P2P data connection</td>
</tr>
<tr>
<td>Bit</td>
<td>Bit</td>
<td>1. Physical</td>
<td>Direct P2P data connection</td>
</tr>
</tbody>
</table>
Network OSI Model

1. Physical
   - Media, signal, and binary transmission
   - Physical addressing
   - Recognizing data
   - TCP, UDP, SCTP, SSL, TLS
   - IP, IPsec, ICMP, IGMP, OSPF
   - Ethernet, 802.11, MAC/LLC/VLAN, ATM, HDP
   - Fibre Channel, Frame Relay, HDLC, PPP, Q.921, Token Ring, ARP
   - RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11

2. Data link
   - Physical layer signaling
   - Media access control (MAC)
   - MAC addresses

3. Network
   - Path determination and logical addressing
   - End-to-end connections and reliability

4. Transport
   - Interhost communication
   - TCP, SIP, RTP
   - RPC - Named pipes
   - TCP, UDP, SCTP, SSL, TLS

5. Session
   - Session establishment
   - Sockets, Session establishment
   - SIP, SIP, RTP
   - RPC - Named pipes

6. Presentation
   - Data representation and encryption
   - Interhost communication

7. Application
   - Network process to application
   - 7. Application
Applications

Kazaa, VoIP, IM, U Tube

Everything on IP

TCP, ICMP

Ethernet, 802.11, Bluetooth

Satellite, Optical

IP on everything

Ossification

Continued Innovations

SDN Tutorial
The Need for a New Network

- The trends driving the networking industry to reevaluate traditional network architecture:
  - The explosion of mobile devices and content
  - Server virtualization
  - Advent of cloud services
- Traditional networks are hierarchical:
  - Make sense for client-server computing
  - Ill-suited to the dynamic computing and storage needs
The key computing trends driving for a new network paradigm:

- Changing traffic patterns:
  - Instead of client-server communication, applications access multiple databases and servers across the entire network

- The “consumerization of IT”:
  - Mobile devices are widely used which drive IT to accommodate these personal devices in a fine-grained manner
The Need for a New Network

- The key computing trends driving for a new network paradigm:
  - The rise of cloud services:
    - Enterprises enthusiastically embraced both **public** and **private** cloud services
  - The hunger for “Big data”:
    - Mega datasets needs massive parallel processing on multiple servers, which need direct connection
    - The constant demand for additional network capacity
    - Maintain any-to-any connectivity
The Need for a New Network

- Limitations of current networking:
  - Complexity leads to stasis:
    - To add or move any device, IT must touch multiple switches, routers, firewalls, etc.
    - Server virtualization has greatly altered assumptions about physical location of hosts
    - Network static nature cannot dynamically adapt changing traffic, application, and user demand
  - Inconsistent policies:
    - Today’s network makes it difficult for IT to apply consistent set of access, security, QoS, and other policies to increasingly mobile users
Limitations of current networking:

- Inability to scale:
  - Network becomes very complex with the addition of thousands of network devices that must be configured and managed
  - Dynamic traffic patterns cannot be handled with manual configuration

- Vendor dependence:
  - Vendors’ equipment product cycles cannot adapt to the rapid changing network architecture in time
  - Lack of standard, open interfaces limit the ability of network operators to tailor the network
To reinvent the Internet
To develop a new network with similar magnitude as today’s Internet but with more demanding and complex design goals and specifications
Built with the usage of emerging new technologies in the area of computer networking
The Internet is *great* at what it does, but..

- **Security** is weak
- **Availability/Reliability** is an issue
- **Instrumentation** is weak
- **Predictability** is weak
- **Manageability** is an issue
- **Mobility** is not well supported
- **Sensing** is not well supported
- **Scalability** is an issue

*Our critical infrastructures cannot rely on it!*

Persistent problems not solvable by incremental improvements to the current Internet

- **New Paradigms** may prove more powerful, providing the basis for a superior Future Internet
**Society Issues**
We increasingly rely on the Internet but are unsure we can trust its security, privacy or resilience.

**Science Issues**
We cannot currently understand or predict the behavior of complex, large-scale networks.

**Innovation Issues**
Substantial barriers to at-scale experimentation with new architectures, services, and technologies.
Software Defined Networking
What is SDN?

• An **architectural approach** that optimizes and **simplifies network** operation by:
  – Decoupling the control plane and the data plane
    • **Control plane**: the system that makes decision about where traffic is sent
    • **Data plane**: the system that forwards traffic to the selected destination

• Evolved from the work done by UC Berkeley and Stanford University (Network managing project)
What is SDN?

• Achieved by employing a point of logically centralized network control (SDN Controller)
  – It facilitates the communication between applications and network elements
  – It exposes and abstracts network functions and operations via programmable interface
  – Gain vendor-independent control over the entire network from a single logical point
What is SDN?

- Today: Closed Boxes, Fully Distributed Protocols
What is SDN?

- SDN was implemented to open it
What is SDN?

- **The Software-defined Network**

1. Open interface to hardware
2. At least one good operating system Extensible, possibly open-source
3. Well-defined open API

Diagram:
- Network Operating System
- Simple Packet Forwarding Hardware
- Apps

22
What is SDN?

Isolated “slices”

Many operating systems, or Many versions

Virtualization or “Slicing” Layer

Open interface to hardware

Simple Packet Forwarding Hardware

SDN Tutorial
Software-Defined Networking (SDN): Data Plane
Network Data Plane

- The part of the router architecture, also called “Forwarding plane”
- It handles incoming datagrams through a series of link-level operations
- Datagram is processed in the data plane by performing lookup in the FIB table programmed by control plane
- Fast path packet processing due to no further learning process needed
Network Data Plane

• One exception to this process when packets cannot be matched to rules
  – Unknown destination detected
  – Packets are sent to router processor where control plane can process

• FIB table can be implemented in varies ways:
  – Software
  – Hardware-accelerated software
  – Hardware
Each cell takes three logic states
- ‘0’, ‘1’, and ‘?’(don’t care)

Fully associative memory: compares input string with all the entries in parallel
- If multiple matches, report index of the first match

Current TCAM technology
- Fast Match Time: 4-8 ns
- Size: 1M
  - 1K entries * 1K bytes per entry
  - 2K entries * 512 bytes per entry
Network Data Plane

• High-performance routers often have multiple distributed forwarding elements
  – Increases performance with parallel processing

• Besides the forwarding decision, the data plane may implement some small features (forwarding features)
  – Access Control List (ACL)
  – Quality of Service (QoS)
  – Policy
Network Data Plane

- The data plane have to do some level of datagram header rewrite

<table>
<thead>
<tr>
<th>Ingress Order of Operation (Generic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decryption (e.g., IF IPSec)</td>
</tr>
<tr>
<td>Input ACL</td>
</tr>
<tr>
<td>Input QOS (e.g., Rate Limit)</td>
</tr>
<tr>
<td>Accounting</td>
</tr>
<tr>
<td>Redirection/Policy Based Routing (PBR)</td>
</tr>
<tr>
<td>Routing</td>
</tr>
</tbody>
</table>
Operation Between Planes

- Two-stage lookups in multislot/card system:
  - 1\textsuperscript{st} stage at ingress identifies the outgoing slot/card
  - 2\textsuperscript{nd} stage at egress performs secondary lookup
  - This can enable an optimization called localization to reduce the egress FIB size
Why Separate?

• **Scalability Issues:**
  – The *service card are limited* to a certain amount of flow state they can support for certain generation of the card
  – The significant *lag between the availability of a new family of processors and new service cards that use that innovation*
  – The *control card memories* have processing limitations based on the generation of the CPU complex

• **Cost**
Why Separate?
SDN Data Plane Technology: OpenFlow
OpenFlow

• A communications protocol that allows the path of network packets through the switches to be determined by software running on multiple routers

• It was originally developed by Stanford University as part of network research
  – Creation of experimental protocols

• Ultimate goal:
  – Replace the functionality of layer 2 and layer 3 protocols completely in commercial switches and routers
OpenFlow

- The key components of the OpenFlow:
  - Separation of the control and data planes
  - Using a standardized protocol between controller and an agent for instantiating state
  - Providing network programmability from a centralized view via API

- It is a set of protocols and an API
  - The controller does nothing without an application program
OpenFlow

Link Aggregation Control Protocol
Rapid Spanning Tree Protocol
Open Shortest Path First
OpenFlow Protocol

• Switch Components:
  – Main components of an OpenFlow switch:
    • Consists of one or more flow tables and a group table, used to perform packet lookup and forwarding
    • An OpenFlow channel to an external controller
    • Controller manages the switch via the OpenFlow protocol
OpenFlow Protocol

• Switch Components:
  – The controller can add, update, and delete **flow entries** in flow tables
  – Each flow table contains a set of flow entries; each flow entry consists of:
    • Match fields, counters, and set of instructions
  – Matching starts at the first flow table and may continue to additional flow tables
  – Flow entries match packets in priority order, with the first matching entry being used
OpenFlow Protocol

Flow Table Entry

OpenFlow 1.0

Flow Entry

Matching Fields | Actions | Stats
--- | --- | ---

Packet counters, byte counters

Examples:
- Forward packet to a port list
- Add/remove/modify VLAN Tag
- Drop packet
- Send packet to the controller

Layer 2

Layer 3

<table>
<thead>
<tr>
<th>Ingress Port</th>
<th>MAC DA</th>
<th>MAC SA</th>
<th>Ether Type</th>
<th>VLAN ID</th>
<th>P-bits</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Protocol</th>
<th>IP DSCP</th>
<th>TCP/UDP src port</th>
<th>TCP/UDP dst port</th>
</tr>
</thead>
</table>

SDN Tutorial
OpenFlow Protocol

Packet Flow Flowchart

Packet In
Start at table 0

Match in table n?

Update counters
Execute instructions:
- update action set
- update packet/match set fields
- update metadata

Goto-Table n?

Table-miss flow entry exists?

Drop packet

Execute action set
Switch Components:

- If a matching entry found, the instructions associate with the specific flow entry are executed
- If no match found, the outcome depends on configuration of the `table-miss` flow entry:
  - The packet may be forwarded to the controller, dropped, or may continue to next flow table
- Instruction associated with each flow entry either contain actions or modify pipeline processing
OpenFlow Protocol

• **Switch Components:**
  – Actions included in instructions describe packet forwarding, packet modification and group table processing
  – Pipeline processing instructions allow packets to be sent to subsequent tables for further processing
  – Pipeline processing stops when the instruction set associated with a matching flow entry doesn’t specify a next table
    • At this point the packet is usually modified and forwarded
OpenFlow Protocol

• Switch Components:
  – Flow entries may forward to a port, usually a physical port, but it could also be a logical port defined by the switch or a reserved port defined by the specification
  – Actions associated with flow entries may also direct packets to a group, which specifies additional processing
  – Groups represent sets of actions for flooding and more complex forwarding semantics (multipath, fast reroute, and link aggregation)
OpenFlow Protocol

• Switch Components:
  – Groups also enable multiple flow entries to forward to a single identifier
  – Group table contains group entries:
    • Each group entry contains a list of action buckets with specific semantics dependent on group type
Example of Nested Flows

From *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud* by William Stallings (0134175395)
Copyright © 2016 Pearson Education, Inc. All rights reserved.
Group Types

From Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud by William Stallings (0134175395) Copyright © 2016 Pearson Education, Inc. All rights reserved.
Goal: Evangelize OpenFlow to vendors

Free membership for all researchers

Whitepaper, OpenFlow Switch Specification, Reference Designs

Licensing: Free for research and commercial use

http://OpenFlowSwitch.org
Mininet

> sudo mn

collectors
switches
hosts
Mininet

• A **network emulator** emulates a collection of end-hosts, switches, routers, and links on a single Linux kernel

• **Uses lightweight virtualization** to make a single system look like a complete network

• Commonly used as an emulation, verification, testing tool, and resource
Mininet

• Represents a shell of a machine that arbitrary programs can be plugged into and run
• The measured performance of a Mininet-hosted network often should approach that of actual (non-emulated) switches, routers, and hosts
• Allows full topologies and packet forwarding customization
Internally, Mininet employs lightweight virtualization features in the Linux kernel, including process groups, CPU bandwidth isolation, and network namespaces, and combines them with link schedulers and virtual Ethernet links.

- **Isolated Hosts.** An emulated host in Mininet is a group of user-level processes moved into a network namespace → Linux network namespaces
- **Emulated Links.** The data rate of each link is enforced by Linux Traffic Control (tc), which has a number of packet schedulers to shape traffic to a configured rate → Linux tc feature
- **Emulated Switches.** Mininet typically uses the default Linux bridge or Open vSwitch running in kernel mode to switch packets across interfaces → Linux Open vSwitch (OVS)
Mininet – Tutorial Overview

OpenFlow tutorial flow:
- set up VM, learn tools, run provided hub
- turn hub into an Ethernet switch
- modify switch to push down flows
- add multiple switch support
- modify switch to handle IP forwarding
- run switch on a real network
- add firewall capabilities to switch
- on your own...
Mininet – Get Started

• The easiest way is to download pre-packaged Mininet/Ubuntu VM
  – Included Mininet, all OpenFlow binaries and tools, and tweaked kernel
  https://bitbucket.org/mininet/mininet-vm-images/downloads

• Download and install a virtualization system
  – Recommend VirtualBox
  https://www.virtualbox.org/wiki/Downloads
Mininet – Get Started

• Import VM:
  – Add the VM and start it up, in the virtualization program you choose
    – VirtualBox:
      • Import the OVF file, select “settings”, and add an additional host-only network adapter
    – VMware:
      • Import the OVF file, then start the VM
  – Qemu/KVM:
    • Convert the VMDK to QCOW2 format first
      `qemu-img convert -O qcow2 filename.vmdk filename.qcow2`
Mininet – Get Started

• VirtualBox:
  – Import
Mininet – Get Started

• VirtualBox:
  – Open downloaded VOF file:
Mininet – Get Started

• VirtualBox:
  – Click next after you chose the location
Mininet – Get Started

- VirtualBox:
  - Review the VM configuration and Import
• VirtualBox:
  – Make sure your VM has two network interfaces:
    • NAT interface:
      – It can use to access the Internet
      – It should be eth0 and have a 10.X IP address
    • Host-only interface:
      – Used to communicate with host machine
      – It should be eth1 with 192.168.X IP address
  • Both interfaces should be configured using DHCP, if not, run:
    – $ sudo dhclient ethX
    – Replacing ethX with the name of downed interface
Mininet – Get Started

- VirtualBox:
  - Select the imported VM and click “Settings”
Mininet – Get Started

• VirtualBox:
  – In the network, add an additional host-only network adapter and click Ok
Mininet – Get Started

- VirtualBox:
  - Log in to VM, use following username and password:
    - mininet-vm login: mininet
    - Password: mininet
  - Command syntax:
    - $: precedes Linux commands that should be typed at the shell prompt
    - mininet>: precedes Mininet commands that should be typed at Mininet’s CLI
    - #: precedes Linux commands that re typed at a root shell prompt
Mac OS and Linux:

- Open a terminal. Run following in terminal:
  
  $ ssh -X [user]@[Guest IP Here]

- Replace [user] with the correct username
- Replace [Guest IP] with the IP you just noded
- Enter the password for your VM image
- Try to start up the X terminal using
  
  $ xterm
Windows:

- In order to use X11 applications such as xterm and wireshark, the Xming server must be running
- Start Xming by double-clicking its icon
- Make an ssh connection with X11 forwarding enabled
Windows:

- To enable X11 forwarding from PuTTY GUI, click PuTTY -> Connection -> SSH -> X11, then click on Forwarding – Enable X11 Forwarding
Windows - Alternative:

– Run X11 in the VM console window:
  • First, log in to the VM in its console window and make sure `apt` is up to date
    ```
sudo apt-get update
    ```
  • Then install the desktop environment of your choice:
    ```
sudo apt-get install xinit <environment>
    ```
  • `<environment>` is your GUI of choice
    – `lxde`: a reasonable compact and fast desktop GUI
    – `flwm`: a smaller but more primitive desktop GUI
    – `ubuntu-desktop`: the full, heavyweight Ubuntu GUI
  • Then you can start X11 in the VM: `startx`
Mininet – Get Started

• Development Environment:
  – OpenFlow Controller:
    • Sits above the OpenFlow interface
    • Executed during the simulation and observe messages being sent
  – OpenFlow Switch:
    • Sits below the OpenFlow interface
    • A user-space software switch
  – dpctl:
    • Command-line utility that sends quick OpenFlow messages
    • Useful for viewing switch port stats
Mininet – Get Started

• Development Environment:
  – Wireshark:
    • General graphical utility for viewing packets
    • Dissector parses OpenFlow messages sent to OpenFlow default port in a readable way
  – iperf:
    • General command-line utility for testing the speed of a single TCP connection
  – cbench:
    • Utility for testing the flow setup rate of OpenFlow controllers
Create the network in VM, enter (SSH):

```
sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

- This tells Mininet to start up 3-host, single switch topology, set the MAC address:
  - Created 3 virtual hosts, each with a separate IP address
  - Created a single OpenFlow software switch in the kernel with 3 ports
  - Connected each virtual host to the switch with a virtual Ethernet cable
  - Set the MAC address of each host equal to its IP
  - Configure the switch to connect to a controller
Mininet – Start Network

• Mininet-specific basic commands:
  – Lists available nodes, run:
    \texttt{mininet}\texttt{> nodes}
  – Lists all available commands, run:
    \texttt{mininet}\texttt{> help}
  – Runs a single command on a node, ex:
    check the IP of a virtual host:
    \texttt{mininet}\texttt{> h1 ifconfig}
  – Running interactive commands and watching debug output:
    \texttt{mininet}\texttt{> xterm h1 h2}
Mininet – Start Network

• dpctl example usage (SSH terminal):
  – It enables visibility and control over a single switch’s flow table, useful for debugging by viewing flow state and flow counters
  – To dump out port state and capabilities:
    
    ```
    $ dpctl show tcp:127.0.0.1:6634
    ```
  – More useful command:
    
    ```
    $ dpctl dump-flows tcp:127.0.0.1:6634
    ```
Ping test:

- Let’s go back to the mininet console and try to ping h2 from h1:

  mininet> h1 ping -c3 h2

  The ping should fail due to the switch flow table is empty, and there is no controller connected to the switch

- Manually install the necessary flows by using dpctl:

  $ dpctl add-flow tcp:127.0.0.1:6634 in_port=1, actions=output:2
  $ dpctl add-flow tcp:127.0.0.1:6634 in_port=2, actions=output:1
In Mininet CLI, open (xterm) terminal for a switch, say s0, and run the following commands in it to enable OpenFlow 1.3 protocol in OVS switches.

```bash
ovs-vsctl set bridge s0 protocols=OpenFlow13
ovs-vsctl set bridge s1 protocols=OpenFlow13
```

Now, add following flow table entries to switch s0 and switch s1 as follows:

```bash
ovs-ofctl -O OpenFlow13 add-flow tcp:127.0.0.1:6634 in_port=1,ip,nw_src=10.0.0.2,nw_dst=10.0.1.2,actions=mod_dl_dst:0A:00:0A:01:00:02,mod_dl_dst:0A:00:0A:FE:00:02,output=2
```
Mininet – Start Network

- Ping test:
  - Run the ping command again and you should get the replies
  - If you didn’t see any ping replies, it might be the case that the flow-entries expired before you start your ping test, the default idle_timeout is 60 seconds
  - We can manually modify the idle_timeout by running:

    $ dpctl add-flow tcp:127.0.0.1:6634 in_port=1,idle_timeout=120,actions=drop

SDN Tutorial
Running Wireshark:

- The VM image includes the OpenFlow Wireshark dissector pre-installed
- To open Wireshark:
  
  $ sudo wireshark &

- Set up a filter for OpenFlow control traffic by typing ‘of’ in Filter box near the top
- Press the apply button to apply the filter to all recorded traffic
Mininet – Start Network

• Start controller and view messages
  – With the Wireshark dissector listening, start the OpenFlow controller (SSH terminal):

  $ controller ptcp:

  – This starts a simple controller that acts as a learning switch without installing any flow-entries

  – You should see number of messages displayed in Wireshark, from the Hello exchange messages and so on
**Mininet – Start Network**

- Start controller and view messages

<table>
<thead>
<tr>
<th>Message</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello</td>
<td>Controller-&gt;Switch</td>
<td>Like TCP handshake, the controller sends its version number to the switch</td>
</tr>
<tr>
<td>Hello</td>
<td>Switch-&gt;Controller</td>
<td>The switch replies with its supported version number</td>
</tr>
<tr>
<td>Features Request</td>
<td>Controller-&gt;Switch</td>
<td>The controller asks to see which ports are available</td>
</tr>
<tr>
<td>Set Config</td>
<td>Controller-&gt;Switch</td>
<td>Controller asks the switch to send flow expirations</td>
</tr>
<tr>
<td>Features Reply</td>
<td>Switch-&gt;Controller</td>
<td>Switch replies with a list of ports, port speed, and supported tables and actions</td>
</tr>
</tbody>
</table>
• Benchmark controller w/iperf:
  – iperf is a command-line tool for checking speeds between two computers
  – In the mininet console, run:
    ```
    mininet> iperf
    ```
  – This command runs an iperf TCP server on one virtual host, then runs an iperf client on a second virtual host, once connected, they transfer packets with each other and report the results