Mininet
An Instant Virtual Network on your Laptop

Kristjon Ciko

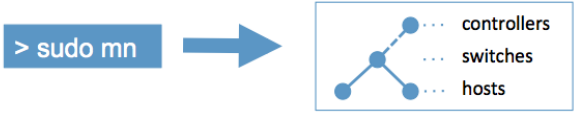
University of Oslo

Mininet Workshop
sponsored by
Norwegian Defence Research Establishment (FFI)

April 30, 2024

Mininet: An Instant Virtual Network on your Laptop

Kristjon Ciko, Ph.D.
Department of Informatics
University of Oslo



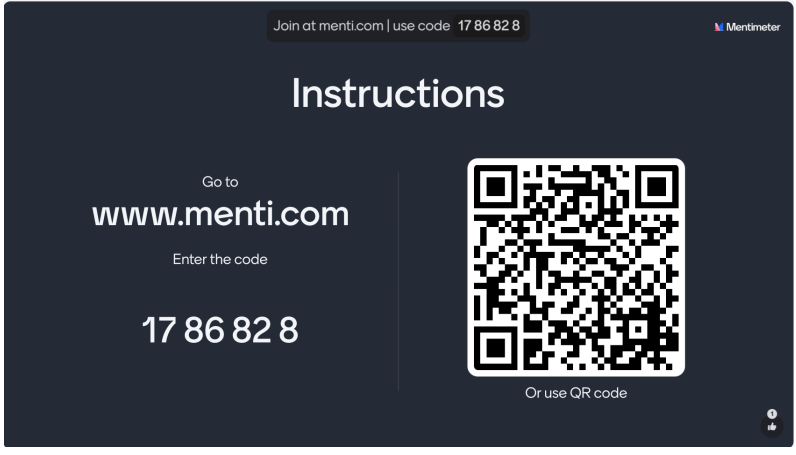
[Photo: https://mininet.org]

UiO : Department of Informatics
University of Oslo

FFI Forsvarets forskningsinstitutt
Norwegian Defence Research Establishment

Who's in this room?!

Who's in this room?!



Goals

Goals

Overarching Goal

Get familiar with Mininet's core functionalities

Goals

By the end, everyone should know:

- ✓ What is Mininet

Goals

By the end, everyone should know:

- ✓ What is Mininet
- ✓ Mininet Command Line Interface (CLI)

Goals

By the end, everyone should know:

- ✓ What is Mininet
- ✓ Mininet Command Line Interface (CLI)
- ✓ Mininet Python API

Goals

By the end, everyone should know:

- ✓ What is Mininet
- ✓ Mininet Command Line Interface (CLI)
- ✓ Mininet Python API
- ✓ Mininet and Software-Defined Networking (SDN)

Agenda

Time	Description
09:00 - 10:00	Introduction to Mininet
10:00 - 10:15	*Short Break*
10:15 - 11:30	Mininet CLI + Python API
11:30 - 12:30	*Lunch Break*
12:30 - 14:00	Mininet and SDN
14:00 - 14:15	*Short Break*
14:15 - 15:30	Mininet and SDN + Extensions

The Workshop will be focused in three main parts

Mininet
Walkthrough

Mininet Python API

Mininet and SDN

The Workshop will be focused in three main parts

Mininet
Walkthrough

Mininet Python API

Mininet and SDN

The Workshop will be focused in three main parts

Mininet Walkthrough

Mininet Python API

Mininet and SDN

The Workshop will be focused in three main parts

Mininet
Walkthrough

Mininet Python API

Mininet and SDN

Mininet Walkthrough

Mininet Python API

Mininet and SDN

What is Mininet?

What is Mininet?

- Network emulator

What is Mininet?

- Network emulator
- Network emulation orchestrator

What is Mininet?

- Network emulator
- Network emulation orchestrator
- Creates realistic virtual network

What is Mininet?

- Network emulator
- Network emulation orchestrator
- Creates realistic virtual network
  - On a single machine

What is Mininet?

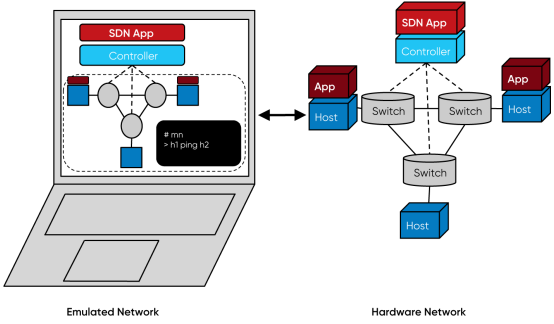
- Network emulator
- Network emulation orchestrator
- Creates realistic virtual network
  - On a single machine
  - With a single command

What is Mininet?

- Network emulator
- Network emulation orchestrator
- Creates realistic virtual network
  - On a single machine
  - With a single command
  - In seconds

What is Mininet?

- Network emulator
- Network emulation orchestrator
- Creates realistic virtual network
  - On a single machine
  - With a single command
  - In seconds



A Mininet network consists of:

Isolated Hosts

A group of user-level processes moved into a network namespace that provide exclusive ownership of interfaces, ports and routing tables.

A Mininet network consists of:

Isolated Hosts

A group of user-level processes moved into a network namespace that provide exclusive ownership of interfaces, ports and routing tables.

Emulated Switches

The default Linux Bridge or the Open vSwitch running in kernel mode is used to switch packets across interfaces. Switches and routers can run in the kernel or in the user space.

A Mininet network consists of:

Isolated Hosts

A group of user-level processes moved into a network namespace that provide exclusive ownership of interfaces, ports and routing tables.

Emulated Links

Each emulated host has its own virtual Ethernet interface(s). Linux Traffic Control *tc* enforces the data rate of each link to shape traffic to a configured rate.

Emulated Switches

The default Linux Bridge or the Open vSwitch running in kernel mode is used to switch packets across interfaces. Switches and routers can run in the kernel or in the user space.

Why use Mininet?

- **It's fast**
  Starting up a simple network takes just a few seconds.

Why use Mininet?

- It's fast
- **You can create custom topologies**
  A single switch, larger Internet-like topologies, a data center, or anything else.

Why use Mininet?

- It's fast
- You can create custom topologies
- **You can run real programs**
  Anything that runs on Linux is available for you to run, from web servers to TCP window monitoring tools to Wireshark.

Why use Mininet?

- It's fast
- You can create custom topologies
- You can run real programs
- **You can run Mininet everywhere**
  On your laptop, on a server, in a VM, on a native Linux box (Mininet is included with Ubuntu 12.10+!), or in the cloud (e.g. Amazon EC2.)

Why use Mininet?

- It's fast
- You can create custom topologies
- You can run real programs
- You can run Mininet everywhere
- **You can share and replicate results**
  Anyone with a computer can run your code.

Why use Mininet?

- It's fast
- You can create custom topologies
- You can run real programs
- You can run Mininet everywhere
- You can share and replicate results
- **You can use it easily**
  You can create and run Mininet experiments by writing simple (or complex if necessary) Python scripts.

Why use Mininet?

- It's fast
- You can create custom topologies
- You can run real programs
- You can run Mininet everywhere
- You can share and replicate results
- You can use it easily
- **Mininet is an open source project**
  `https://github.com/mininet`

Installation and Setup

1. Mininet VM Installation

Download a Mininet VM Image from
`https://github.com/mininet/mininet/releases/`

Installation and Setup

1. Mininet VM Installation

2. Native Installation from Source

```
# Get the source code
git clone https://github.com/mininet/mininet
# Run the following command to install Mininet
mininet/util/install.sh [options]
```

Installation and Setup

1. Mininet VM Installation

2. Native Installation from Source

3. **Installation from Packages**

```
# Install the base Mininet package
sudo apt install mininet
# Test Open vSwitch
sudo mn --switch ovsbr --test pingall
# Make sure Open vSwitch is installed
sudo apt-get install openvswitch-switch
sudo service openvswitch-switch start
```

Hands-on Demo Session

Instructions at:

`https://github.com/kristjoc/org-mininet/`

Mininet Walkthrough

Mininet Python API

Mininet and SDN

Mininet Python API

Defining Custom Topologies through Python API

```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def build( self ):
        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )

        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Mininet Python API

1. Low-level API

The low-level API consists of the base node and link classes (such as *Host*, *Switch*, and *Link* and their subclasses) which can actually be instantiated individually and used to create a network.

Mininet Python API

1. Low-level API: Nodes and Links

```
h1 = Host( 'h1' )
h2 = Host( 'h2' )
s1 = OVSSwitch( 's1', inNamespace=False )
c0 = Controller( 'c0', inNamespace=False )
Link( h1, s1 )
Link( h2, s1 )
h1.setIP( '10.1/8' )
h2.setIP( '10.2/8' )
c0.start()
s1.start( [ c0 ] )
print( h1.cmd( 'ping -c1', h2.IP() ) )
s1.stop()
c0.stop()
```

Mininet Python API

1. Low-level API

2. Mid-level API

The mid-level API adds the *Mininet* object which serves as a container for nodes and links. It provides a number of methods (such as *addHost()*, *addSwitch()*, and *addLink()*) for adding nodes and links to a network, as well as network configuration, startup and shutdown (notably *start()* and *stop()*.)

Mininet Python API

1. Low-level API

2. Mid-level API: Network object

```
net = Mininet()
h1 = net.addHost( 'h1' )
h2 = net.addHost( 'h2' )
s1 = net.addSwitch( 's1' )
c0 = net.addController( 'c0' )
net.addLink( h1, s1 )
net.addLink( h2, s1 )
net.start()
print( h1.cmd( 'ping -c1', h2.IP() ) )
CLI( net )
net.stop()
```

Mininet Python API

1. Low-level API

2. Mid-level API

3. High-level API

The high-level API adds a topology template abstraction, the *Topo* class, which provides the ability to create reusable, parametrized topology templates. These templates can be passed to the *mn* command (via the *–custom* option) and used from the command line.

```
sudo mn --custom custom_example.py --topo mytopo
```

Mininet Python API

1. Low-level API

2. Mid-level API

3. High-level API: Topology templates

```
class SingleSwitchTopo( Topo ):
    def build( self, count=1 ):
        hosts = [ self.addHost( 'h%d' % i )
                  for i in range( 1, count + 1 ) ]
        s1 = self.addSwitch( 's1' )
        for h in hosts:
            self.addLink( h, s1 )

net = Mininet( topo=SingleSwitchTopo( 3 ) )
net.start()
CLI( net )
net.stop()
```

Mininet Python API

Mininet API Documentation

```
python
>>> from mininet.node import Host
>>> help(Host.IP)
Help on method IP in module mininet.node:

IP(self, intf=None) unbound mininet.node.Host method
        Return IP address of a node or specific interface.
```

Hands-on Demo Session

Instructions at:

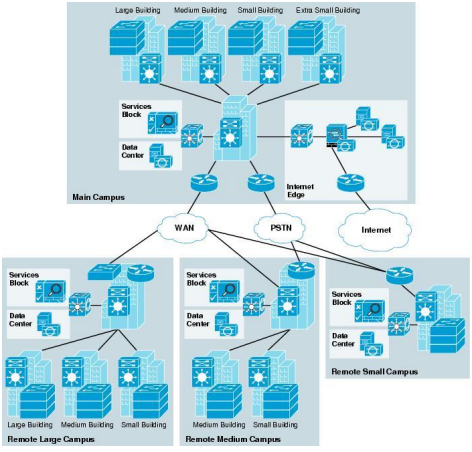
```
https://github.com/kristjoc/org-mininet/
```

Mininet Walkthrough

Mininet Python API

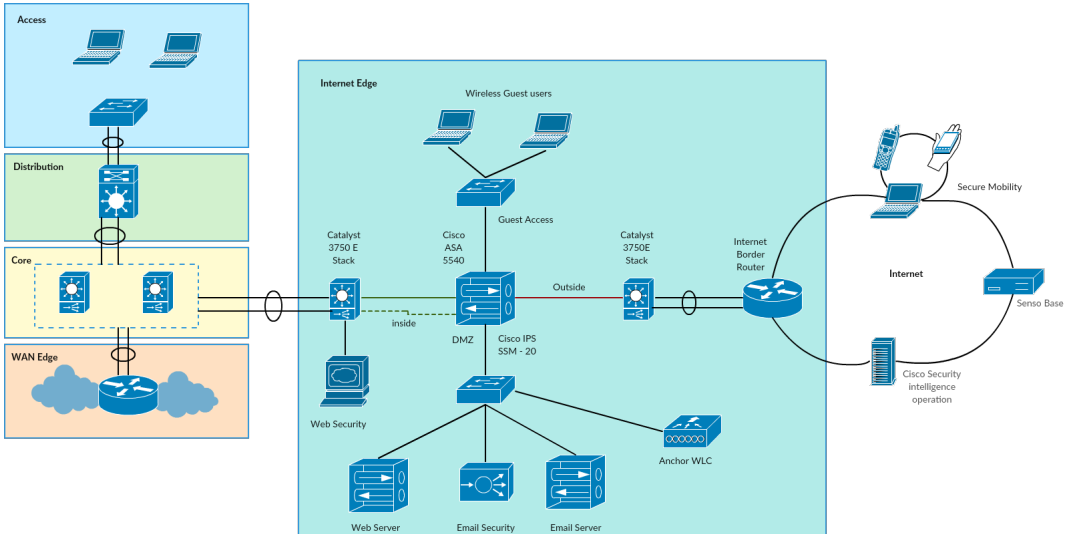
Mininet and SDN

First, networks are enormous in size



[Photo: Cisco]

Second, networks are highly heterogeneous



Second, networks are highly heterogeneous

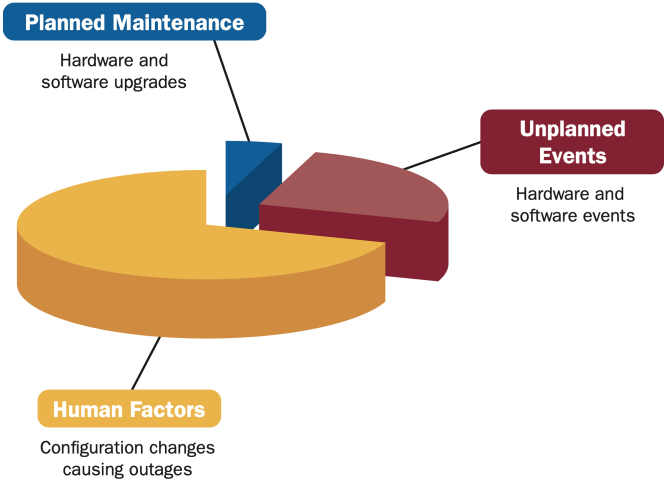
Appliance type	Number
Firewalls	166
NIDS	127
Conferencing/Media gateways	110
Load balancers	67
Proxy caches	66
VPN devices	45
WAN optimizers	44
Voice gateways	11
Middleboxes total	636
Routers	≈ 900

[Source: Sekar et al, Design and Implementation of a Consolidated Middlebox Architecture, NSDI 12]

Third, networks are very complex to manage

[Photo: Google Images]

Third, networks are very complex to manage



[Photo: Juniper]

Third, networks are very complex to manage

	Misconfig.	Overload	Physical/Electric
Firewalls	67.3%	16.3%	16.3%
Proxies	63.2%	15.7%	21.1%
IDS	54.5%	11.4%	34%

[Source: Sherry et al, Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service, SIGGCOM 12]

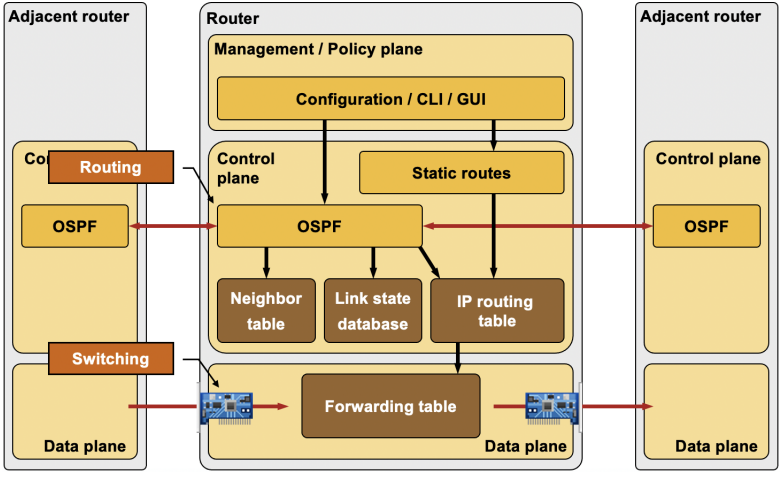
What is Software-Defined Networking (SDN)?

What is SDN?

Physical separation of the network control plane from the forwarding plane

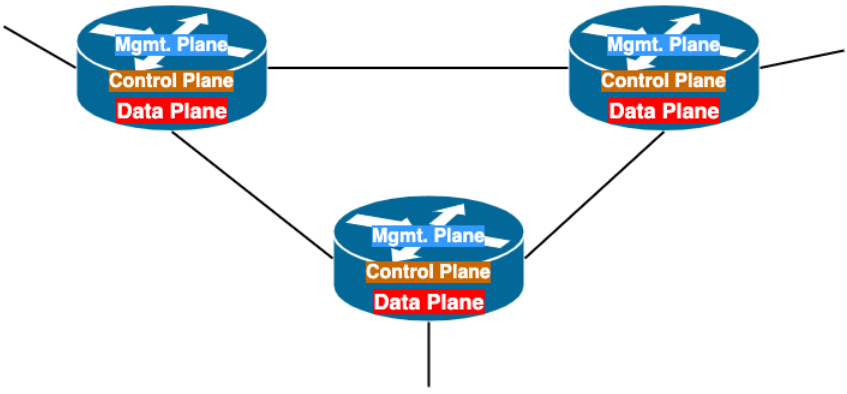
The control plane controls several devices and is directly programmable

Network planes (Mgmt., Control, and Data planes)

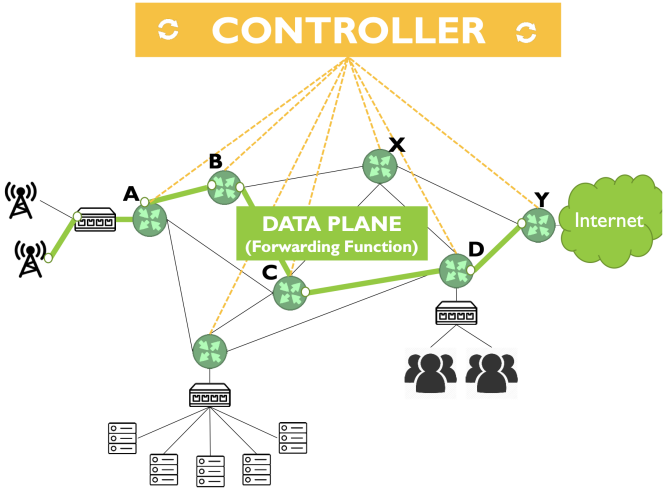


[Photo: ipSpace.net]

In traditional networking, Control and Data planes reside within the physical device



SDN: Separation of Control and Data planes



[Photo: TelecomTutorial.org]

What are the benefits of SDN?

What are the benefits of SDN?

- Directly programmable

What are the benefits of SDN?

- Directly programmable
- Agile

What are the benefits of SDN?

- Directly programmable
- Agile
- Centrally managed

What are the benefits of SDN?

- Directly programmable
- Agile
- Centrally managed
- Open Standard based

What are the benefits of SDN?

- Directly programmable
- Agile
- Centrally managed
- Open Standard based
- Vendor-neutral

What are the challenges of SDN?

What are the challenges of SDN?

- Standardization and Adoption

What are the challenges of SDN?

- Standardization and Adoption
- Reliability

What are the challenges of SDN?

- Standardization and Adoption
- Reliability
- Performance

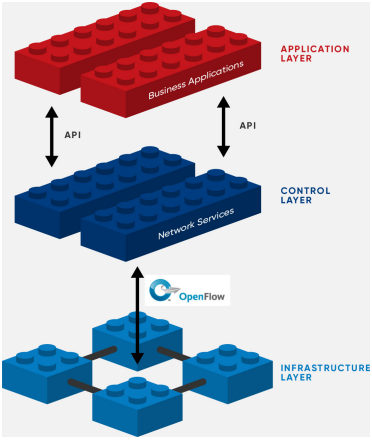
What are the challenges of SDN?

- Standardization and Adoption
- Reliability
- Performance
- Scalability

What are the challenges of SDN?

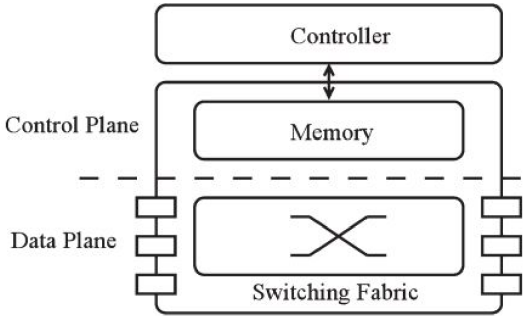
- Standardization and Adoption
- Reliability
- Performance
- Scalability
- Security

SDN Architecture



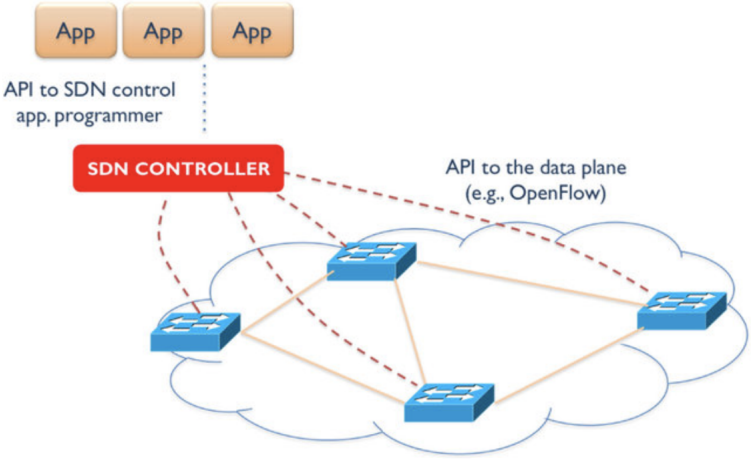
[Photo: Open Networking Foundation]

SDN Infrastructure Layer



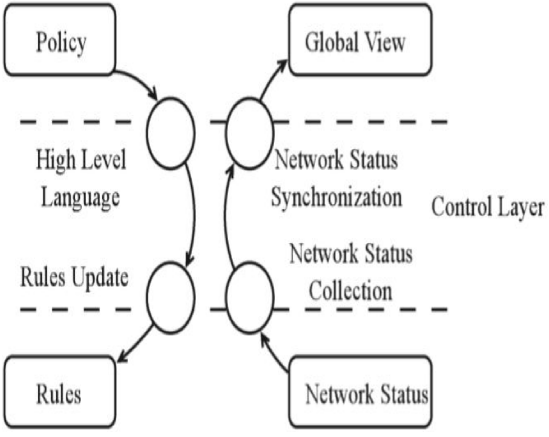
[Source: Xia et al, A Survey on Software-Defined Networking, IEEE COMST 14]

SDN Control Layer



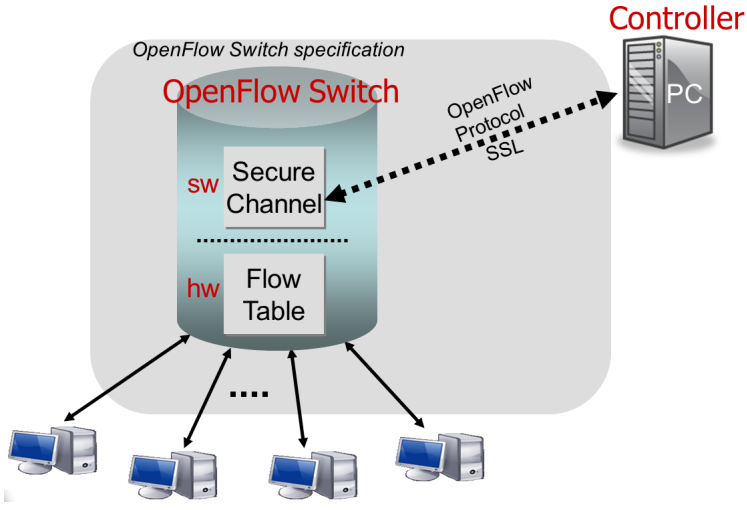
[Source: ResearchGate]

SDN Control Layer



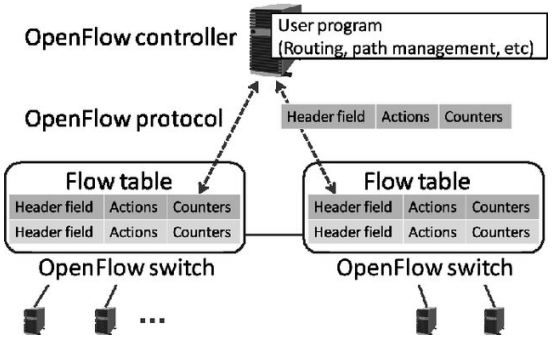
[Source: Xia et al, A Survey on Software-Defined Networking, IEEE COMST 14]

OpenFlow Protocol



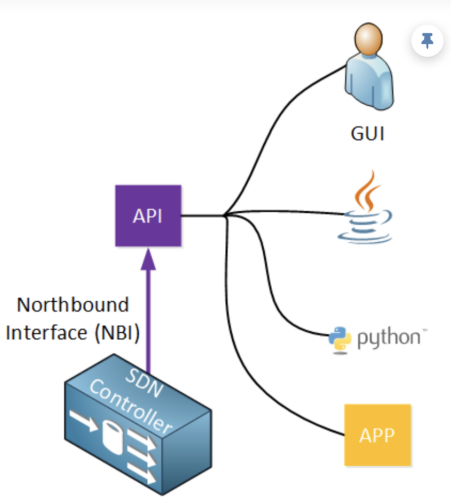
[Source: slideshare.net]

OpenFlow Protocol



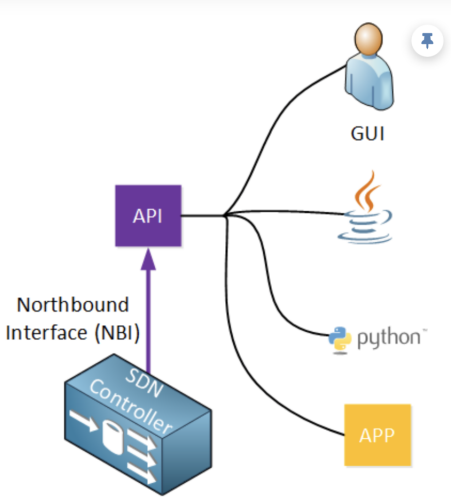
[Source: Suzuki et al, A Survey on OpenFlow Technologies, IEICE ToC, 14]

SDN Application Layer



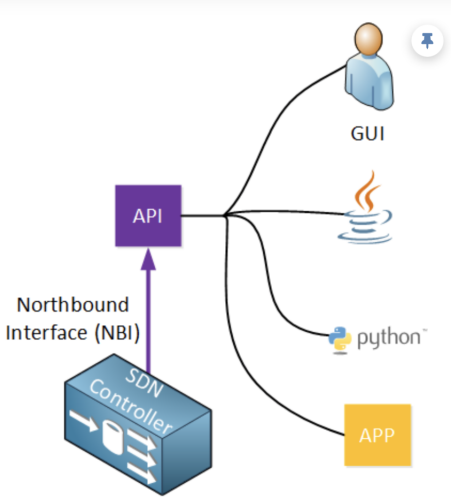
- List information from all network devices in your network

SDN Application Layer



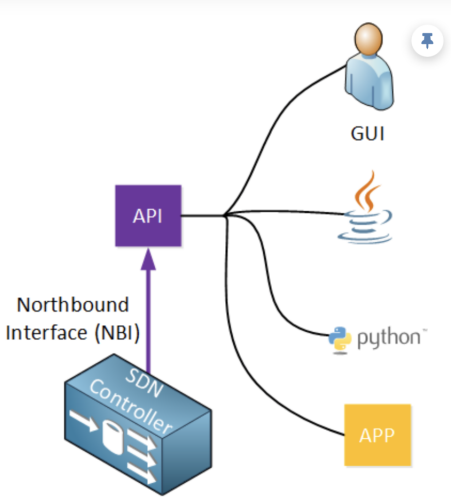
- List information from all network devices in your network
- Show the status of all physical interfaces in the network

SDN Application Layer



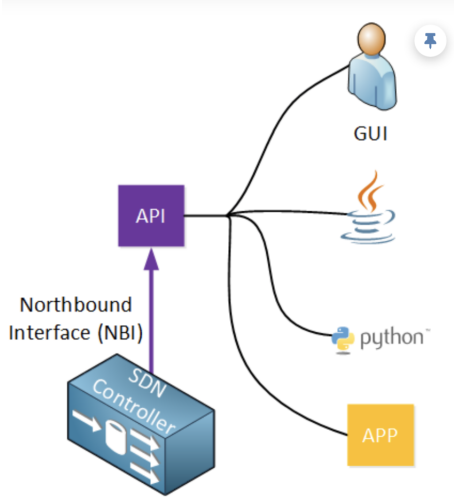
- List information from all network devices in your network
- Show the status of all physical interfaces in the network
- Add a new VLAN on all your switches

SDN Application Layer



- List information from all network devices in your network
- Show the status of all physical interfaces in the network
- Add a new VLAN on all your switches
- Show the topology of your entire network

SDN Application Layer



- List information from all network devices in your network
- Show the status of all physical interfaces in the network
- Add a new VLAN on all your switches
- Show the topology of your entire network
- Automatically configure IP addresses, routing, and access-lists when a new virtual machine is created

Hands-on Demo Session

Instructions at:

```
https://github.com/kristjoc/org-mininet/
```

Mininet extensions

MaxiNet

MaxiNet extends the Mininet emulation environment to span the emulation across several physical machines in order to emulate very large software-defined networks.

Mininet extensions

MaxiNet

DistriNet

Distrinet is a distributed network emulator that provides a way to distribute Mininet over multiple hosts. Distrinet uses the same API as Mininet and it is fully compatible with Mininet programs.

Mininet extensions

MaxiNet

DistriNet

Containernet

Containernet is a fork of the Mininet network emulator and allows to use Docker containers as hosts in emulated network topologies.

Installation

```
# First install Ansible:
sudo apt-get install ansible
# Then clone the repository:
git clone https://github.com/containernet/containernet.git
# Finally run the Ansible playbook to install required dependencies:
sudo ansible-playbook -i "localhost," -c local \
                      containernet/ansible/install.yml
```

Installation

Get started

```
# First install Ansible:
sudo apt-get install ansible
# Then clone the repository:
git clone https://github.com/containernet/containernet.git
# Finally run the Ansible playbook to install required dependencies:
sudo ansible-playbook -i "localhost," -c local \
                      containernet/ansible/install.yml
```

Containernet

Installation

Get started

Build container examples

```
# Build containers
sudo bash containernet/examples/example-containers/build.sh
```

Containernet

Installation

Get started

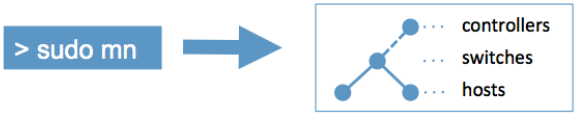
Build container examples

Run a basic example

```
# Start an example topology
sudo python3 examples/containernet_example.py
```

In conclusion: Mininet can be your Swiss army knife for experimenting with networks and SDN.

Questions?



UiO : Department of Informatics
University of Oslo

FFI Forsvarets forskningsinstitutt
Norwegian Defence Research Establishment

In conclusion: Mininet can be your Swiss army knife for experimenting with networks and SDN.

Questions?