Computer Networks

Introduction to Mininet

What is Mininet

Mininet is a tool for software-defined networks. It is an emulator of a network and it is used to visualize the switches and application of software-defined networks in a virtualized environment.

Mininet Setup Prerequisites (Mac OS)

Download Multipass: https://multipass.run/docs/installing-on-macos

Download Quartz: https://www.xquartz.org/

If you are using Homebrew, **brew install --cask multipass**

Mininet: Download & Install (Mac OS)

You can find steps to setup mininet in the project description :

https://courses.cs.washington.edu/cse461/23au/assignments/multipass.html

Additional setup details and debugging instructions:
https://blog.sflow.com/2020/11/multipass.html

Mininet Setup Prerequisites (Windows)

Download Multipass: https://multipass.run/docs/installing-on-windows

Download VM VirtualBox: https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html

If you are using Windows 10 Enterprise or Pro, you can use Hyper-v instead of VirtualBox.

The mininet installation once your VM is properly installed is the same as on MacOS.

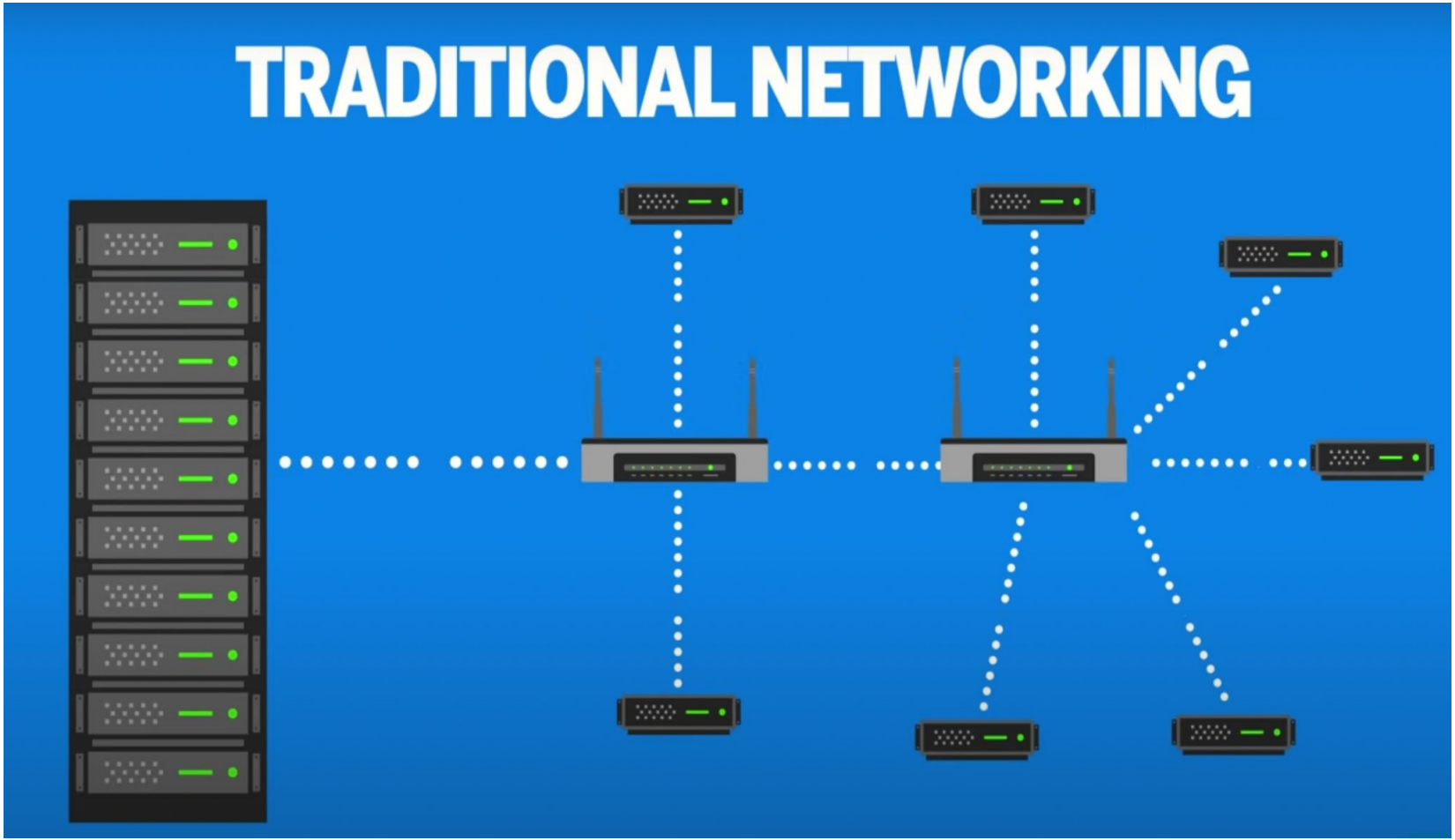
Some Basic Commands

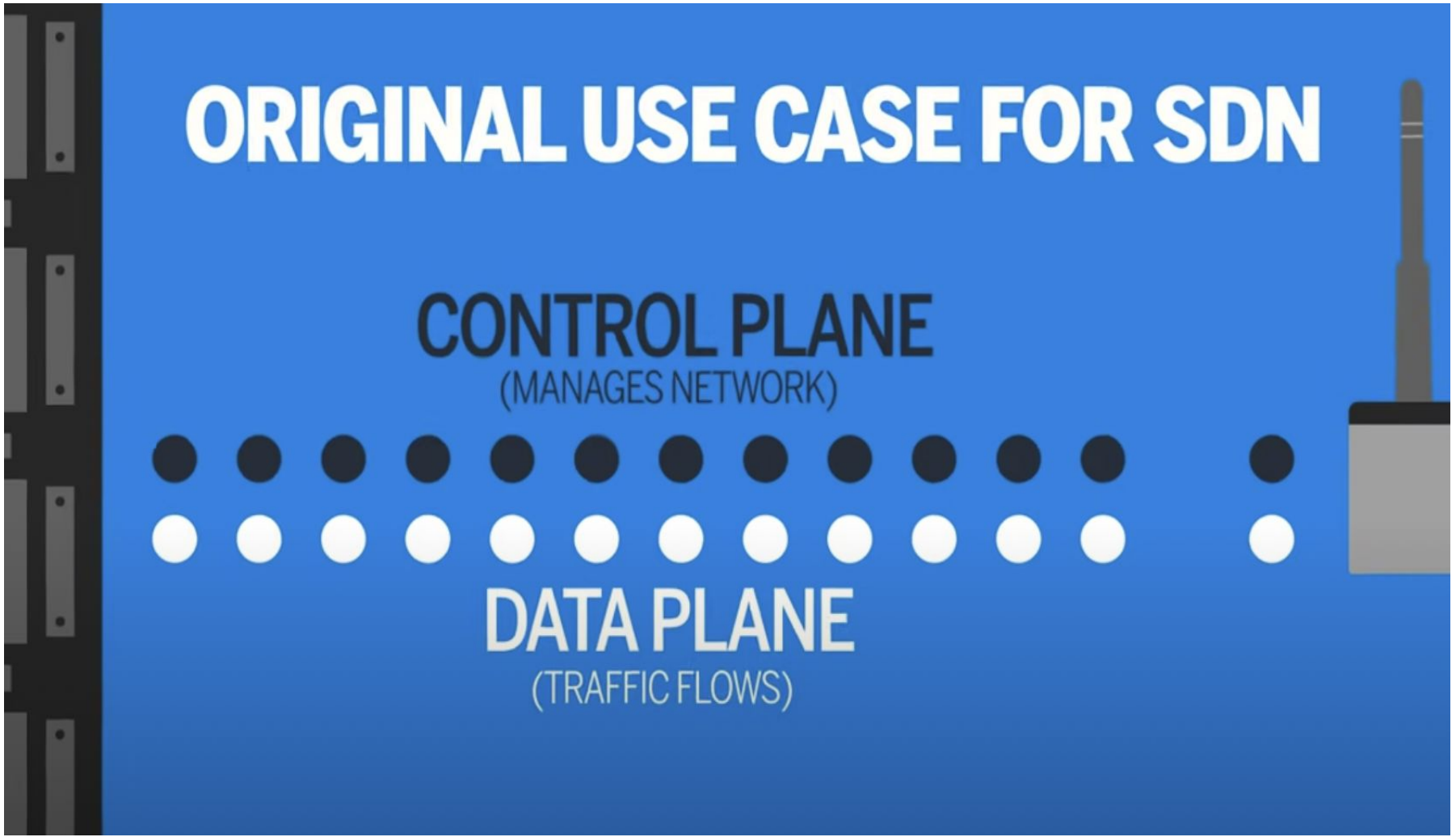
To setup a basic topology use, sudo -E mn

- Display Mininet CLI commands:
  mininet> help
- Display nodes:
  mininet> nodes
- Display links:
  mininet> links
- Dump information about all nodes:
  mininet> dump
- Display Interfaces
  mininet> intfs
- Ping between 2 hosts
  mininet> h1 ping h2
- Ping all the hosts
  mininet> pingall

Software Defined Network (SDN)

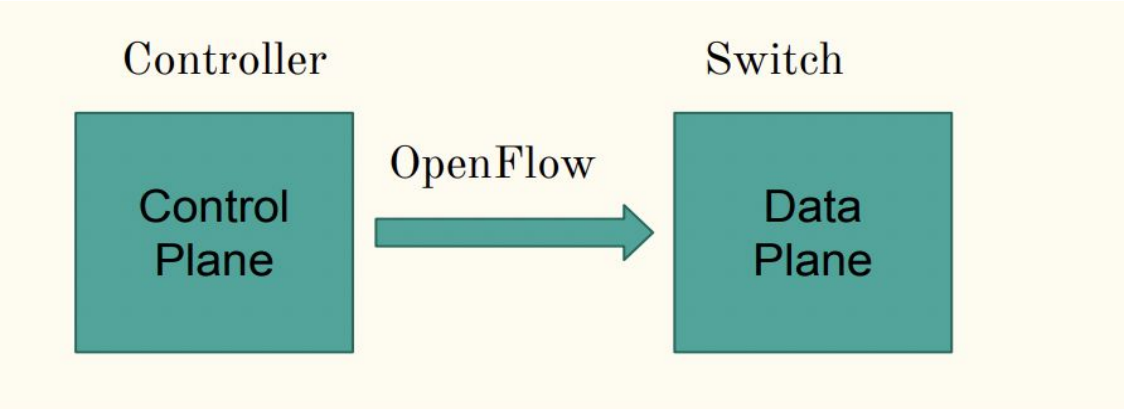
SDN is attributed with “Bringing the tenets of virtualization to networking”.


TRADITIONAL NETWORKING


ORIGINAL USE CASE FOR SDN
CONTROL PLANE
(MANAGES NETWORK)
DATA PLANE
(TRAFFIC FLOWS)

Software Defined Network (SDN)

(SDN) technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring making it more like cloud computing than traditional network management.



Data Transfer Between Hosts

- **Using SimpleHTTPServer and wget :**
  - ***SimpleHTTPServer*** is a python based server application for hosting the files in a system to anyone with the system's IP address.
  - ***wget*** is used to obtain the file from the server.
  - *To run a server:* python3 -m http.server 80
  - *To run a client:* wget -o - [server_ip]

- **Iperf** is a widely used tool for network performance measurement and tuning. It helps in transferring actual data between the hosts.
  - **TCP transfer**
    - *Server:* iperf -s &
    - *Client:* iperf -c [server_ip]
  - **UDP transfer**
    - *Server:* iperf -s -u
    - *Client:* iperf -c [server_ip] -u -b [bandwidth_value]

Topologies in Mininet

- Single
  - sudo -E mn --topo single,2
- Reversed
  - sudo -E mn --topo reversed,2
- Linear
  - sudo -E mn --topo linear,3,2
- Tree
  - sudo -E mn --topo tree,3,2

Custom topologies in Mininet

- Create a python file : http://xuyansen.work/create-a-custom-topology-in-mininet/
- Run the topology using the command : sudo -E mn --custom project.py --topo=project

Mininet + Pox Controller

- Pox: A Python-based SDN controller platform geared towards research and education. You will be using it to set up rules on the Mininet switches.
- Create a new VM with mininet installed and do
  - git clone https://github.com/noxrepo/pox
  - sudo ~/pox/pox.py forwarding.l2_pairs info.packet_dump samples.pretty_log log.level --DEBUG
- Go back to your original VM and do
  - sudo -E mn --controller=remote,ip=<pox_vm_ip>,port=6633

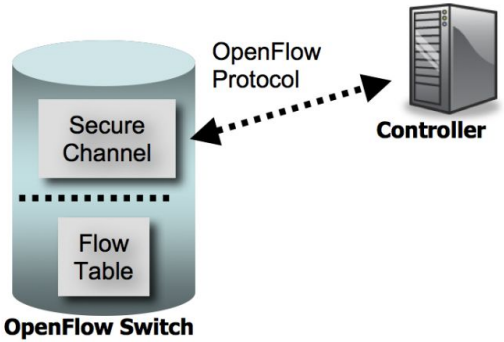
Mininet + Pox Controller

Some helpful links:

- https://www.brianlinkletter.com/2015/04/using-the-pox-sdn-controller/
- https://www.comp.nus.edu.sg/~tbma/teaching/cs4226y16_past/tutorial-Mininet-POX.pdf
- https://noxrepo.github.io/pox-doc/html/

OpenFlow

- OpenFlow is a protocol / standard established by the Open Networking Foundation
- The standard defines the capability in switches to remotely establish rules in the flow table to manage incoming packets
- Each flow rule has three components: the fields, the counters, the action



Flow Rules

- The field component determines which incoming packages match with the rule: fields you might find useful are ethertype, IP source address, and IP protocol
- OpenFlow requires the forward actions to be implemented as well as actions that modify the IP and ether headers (the specifics can be found in the 1.0 specification)

Field	Bits	When applicable	Notes
Ingress Port	(Implementation dependent)	All packets	Numerical representation of incoming port, starting at 1.
Ethernet source address	48	All packets on enabled ports	
Ethernet destination address	48	All packets on enabled ports	
Ethernet type	16	All packets on enabled ports	An OpenFlow switch is required to match the type in both standard Ethernet and 802.2 with a SNAP header and OUI of 0x000000. The special value of 0x05FF is used to match all 802.3 packets without SNAP headers.
VLAN id	12	All packets of Ethernet type 0x8100	
VLAN priority	3	All packets of Ethernet type 0x8100	VLAN PCP field
IP source address	32	All IP and ARP packets	Can be subnet masked
IP destination address	32	All IP and ARP packets	Can be subnet masked
IP protocol	8	All IP and IP over Ethernet, ARP packets	Only the lower 8 bits of the ARP opcode are used
IP ToS bits	6	All IP packets	Specify as 8-bit value and place ToS in upper 6 bits.
Transport source port / ICMP Type	16	All TCP, UDP, and ICMP packets	Only lower 8 bits used for ICMP Type
Transport destination port / ICMP Code	16	All TCP, UDP, and ICMP packets	Only lower 8 bits used for ICMP Code

OpenFlow + Pox

- In Pox, the controller will be handed its connections to the switches on startup, from which you can create and send OpenFlow Flow Rules
- Each connection object will also have the DPID of the switch it connects to, which can be used to setup different switches in unique ways
- Other method hooks including PacketIn defined by Pox interface

```
def __init__ (self, connection):
  # Keep track of the connection to the switch
  # send it messages!
  self.connection = connection

  # This binds our PacketIn event listener
  connection.addListeners(self)

  # Define some sort of OpenFlow Flow Rule
  new_fm = of.ofp_flow_mod()
  # Set Fields and Actions of new rule
  new_fm.match = of.ofp_match(dl_type = 0x0800)

  self.connection.send(new_fm)
```

Mininet + Openflow

Some helpful links:

- https://github.com/noxrepo/pox-doc/blob/master/include/openflow.rst
- https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf (First ten pages)

Start early!

It is not hard, but you will probably spend a lot time looking for and reading documentation...

Resources About Mininet & Pox

Mininet:

- https://github.com/mininet/mininet/wiki/Documentation
- https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#creating

Pox Wiki and API docs:

- https://noxrepo.github.io/pox-doc/html/#id97

Pox OpenFlow Tutorials:

- https://github.com/mininet/openflow-tutorial/wiki/Create-a-Learning-Switch#Controller_Choice_POX_Python
- https://haryachyy.wordpress.com/2014/06/14/learning-pox-openflow-controller-proactive-approach/