Mininet & POX Controller Tutorial

Instructor: Dr. Irfan Ali
Presenter: Müge Erel-Özçevik

Department of Computer Engineering,
Istanbul Technical University,
Turkey
Email:erelmu@itu.edu.tr

18.04.2017

Outline

Outline

Platforms for Network/Systems

Platform	Advantages	Disadvantages
Hardware Testbed	fast accurate: "ground truth"	expensive shared resource? hard to reconfigure hard to change hard to download
Simulator	inexpensive, flexible detailed (or abstract!) easy to download virtual time (can be "faster" than reality)	may require app changes might not run OS code detail != accuracy may not be "believable" may be slow/non-interactive
Emulator	**inexpensive, flexible** **real code** **reasonably accurate** **easy to download** **fast/interactive usage**	slower than hardware experiments may not fit possible inaccuracy from multiplexing

Figure: Platforms for Network/Systems

Mininet Architecture

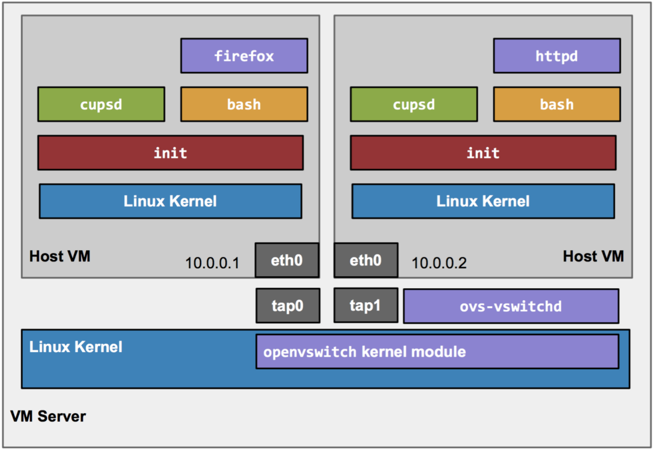


Figure: An example of Mininet Architecture for two hosts

Outline

Mininet Installation

To get started install these softwares on your host machine:

- Install Vagrant, it is a wrapper around virtualization softwares like VirtualBox, VMWare etc.: http://www.vagrantup.com/downloads
- Install VirtualBox, this would be your VM provider: https://www.virtualbox.org/wiki/Downloads
- Install Git, it is a distributed version control system: https://git-scm.com/downloads
- Install X Server and SSH capable terminal
  - Linux comes pre-installed with X server and Gnome terminal + SSH (buitlin)
  - »sudo apt-get install ubuntu-desktop

Mininet Installation

To get started install these softwares on your host machine:

- Install Vagrant, it is a wrapper around virtualization softwares like VirtualBox, VMWare etc.: http://www.vagrantup.com/downloads
- Install VirtualBox, this would be your VM provider: https://www.virtualbox.org/wiki/Downloads
- Install Git, it is a distributed version control system: https://git-scm.com/downloads
- Install X Server and SSH capable terminal
    - Linux comes pre-installed with X server and Gnome terminal + SSH (buitlin)
    - »sudo apt-get install ubuntu-desktop

Mininet Installation

To get started install these softwares on your host machine:

- Install Vagrant, it is a wrapper around virtualization softwares like VirtualBox, VMWare etc.: http://www.vagrantup.com/downloads
- Install VirtualBox, this would be your VM provider: https://www.virtualbox.org/wiki/Downloads
- Install Git, it is a distributed version control system: https://git-scm.com/downloads
- Install X Server and SSH capable terminal
  - Linux comes pre-installed with X server and Gnome terminal + SSH (buitlin)
  - »sudo apt-get install ubuntu-desktop

Mininet Installation

To get started install these softwares on your host machine:

- Install Vagrant, it is a wrapper around virtualization softwares like VirtualBox, VMWare etc.: http://www.vagrantup.com/downloads
- Install VirtualBox, this would be your VM provider: https://www.virtualbox.org/wiki/Downloads
- Install Git, it is a distributed version control system: https://git-scm.com/downloads
- Install X Server and SSH capable terminal
  - Linux comes pre-installed with X server and Gnome terminal + SSH (buitlin)
  - »sudo apt-get install ubuntu-desktop

Mininet Installation

To get started install these softwares on your host machine:

- Install Vagrant, it is a wrapper around virtualization softwares like VirtualBox, VMWare etc.: http://www.vagrantup.com/downloads
- Install VirtualBox, this would be your VM provider: https://www.virtualbox.org/wiki/Downloads
- Install Git, it is a distributed version control system: https://git-scm.com/downloads
- Install X Server and SSH capable terminal
  - Linux comes pre-installed with X server and Gnome terminal + SSH (buitlin)
  - »sudo apt-get install ubuntu-desktop

Mininet Installation

To get started install these softwares on your host machine:

- Install Vagrant, it is a wrapper around virtualization softwares like VirtualBox, VMWare etc.: http://www.vagrantup.com/downloads
- Install VirtualBox, this would be your VM provider: https://www.virtualbox.org/wiki/Downloads
- Install Git, it is a distributed version control system: https://git-scm.com/downloads
- Install X Server and SSH capable terminal
  - Linux comes pre-installed with X server and Gnome terminal + SSH (buitlin)
  - »sudo apt-get install ubuntu-desktop

Outline

Help

»sudo mn -h

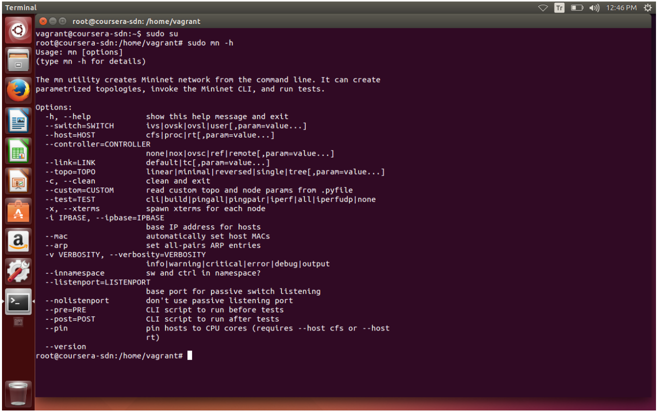


Figure: Calling Mininet Help

Linear topology

(»sudo mn - - topo=linear,5) (»nodes) (»net) (»dump)

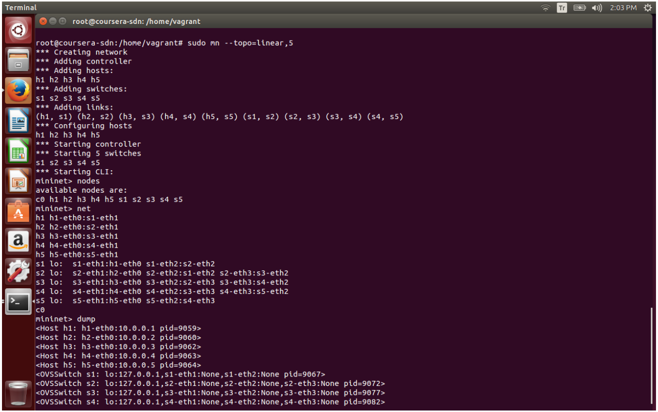


Figure: Examining Linear Topology with 5 OpenFlow switches

Flow table status

»dpctl dump-flows

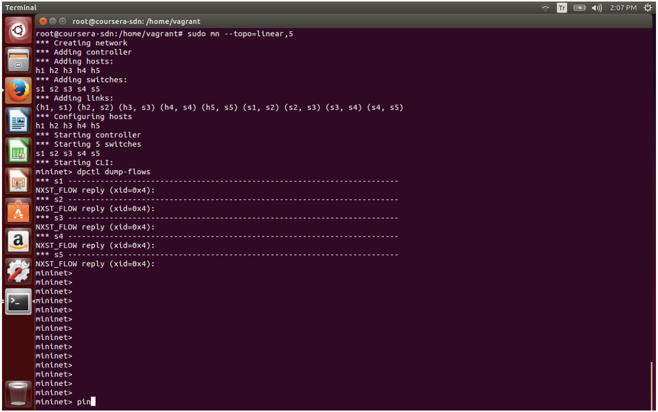


Figure: Empty flow tables

Pingall

»pingall

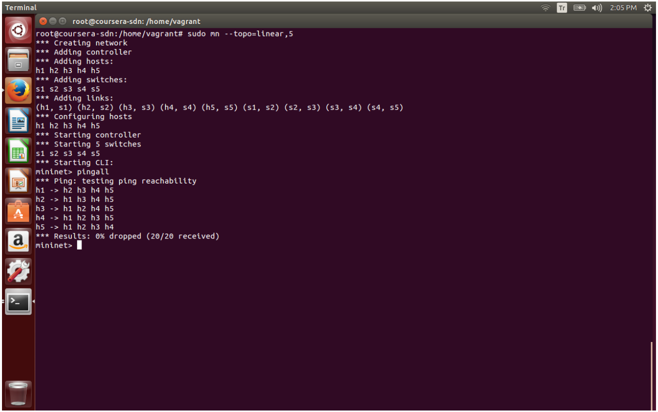


Figure: Examining Linear Topology with 5 OpenFlow switches

Flow tables after iperf

»iperf h1 h2

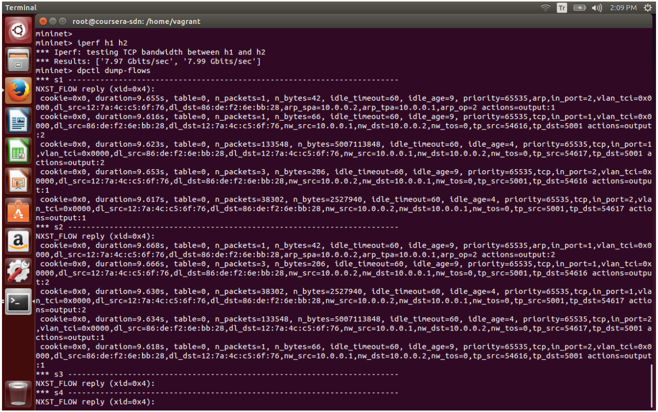
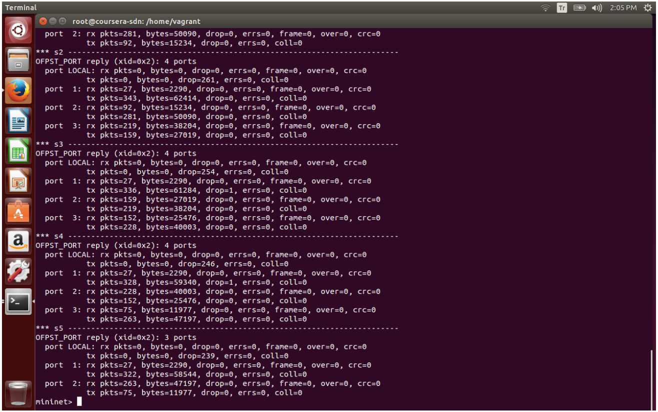


Figure: Flow tables are filled via TCP flow entries after iperf

Port status

»dpctl dump-ports



```
Terminal                                             2:05 PM
root@coursera-sdn: /home/vagrant
  port  2: rx pkts=281, bytes=50090, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=92, bytes=15234, drop=0, errs=0, coll=0
*** s2 ------------------------------------------------------------------------
OFPST_PORT reply (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=261, errs=0, coll=0
  port  1: rx pkts=27, bytes=2290, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=343, bytes=62414, drop=0, errs=0, coll=0
  port  2: rx pkts=92, bytes=15234, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=281, bytes=50090, drop=0, errs=0, coll=0
  port  3: rx pkts=219, bytes=38204, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=159, bytes=27019, drop=0, errs=0, coll=0
*** s3 ------------------------------------------------------------------------
OFPST_PORT reply (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=254, errs=0, coll=0
  port  1: rx pkts=27, bytes=2290, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=336, bytes=61284, drop=1, errs=0, coll=0
  port  2: rx pkts=159, bytes=27019, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=219, bytes=38204, drop=0, errs=0, coll=0
  port  3: rx pkts=152, bytes=25476, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=228, bytes=40003, drop=0, errs=0, coll=0
*** s4 ------------------------------------------------------------------------
OFPST_PORT reply (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=246, errs=0, coll=0
  port  1: rx pkts=27, bytes=2290, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=328, bytes=59340, drop=1, errs=0, coll=0
  port  2: rx pkts=228, bytes=40003, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=152, bytes=25476, drop=0, errs=0, coll=0
  port  3: rx pkts=75, bytes=11977, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=263, bytes=47197, drop=0, errs=0, coll=0
*** s5 ------------------------------------------------------------------------
OFPST_PORT reply (xid=0x2): 3 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=239, errs=0, coll=0
  port  1: rx pkts=27, bytes=2290, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=322, bytes=58544, drop=0, errs=0, coll=0
  port  2: rx pkts=263, bytes=47197, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=75, bytes=11977, drop=0, errs=0, coll=0
mininet>
```

Figure: Port status

Outline

POX controller default running

- Run Controller with:
  - »python pox.py
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables
  - Destination Host Unreachable while try to ping it

POX controller default running

- Run Controller with:
  - »python pox.py
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables
  - Destination Host Unreachable while try to ping it

POX controller default running

- Run Controller with:
  - »python pox.py
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables
  - Destination Host Unreachable while try to ping it

POX controller default running

- Run Controller with:
  - »python pox.py
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables
  - Destination Host Unreachable while try to ping it

POX controller default running

- Run Controller with:
  - »python pox.py
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables
  - Destination Host Unreachable while try to ping it

POX controller default running

- Run Controller with:
  - »python pox.py
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables
  - Destination Host Unreachable while try to ping it

POX controller default running

- Run Controller with:
  - »python pox.py
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables
  - Destination Host Unreachable while try to ping it

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - It installs exact-match rules for each flow
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with layer2 learning

- Run Controller with:
  - »python pox.py forwarding.l2_learning
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633
- Results:
  - Empty flow tables initially
  - When it is tried to ping from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable

POX controller with spanning tree in DEBUG mode

- Run Controller with:
  - »python pox.py forwarding.l2_learning openflow.spanning_tree -no-flood -hold-down log.level -DEBUG samples.pretty_log openflow.discovery host_tracker info.packet_dump
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633

POX controller with spanning tree in DEBUG mode

- Run Controller with:
  - »python pox.py forwarding.l2_learning openflow.spanning_tree -no-flood -hold-down log.level -DEBUG samples.pretty_log openflow.discovery host_tracker info.packet_dump
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633

POX controller with spanning tree in DEBUG mode

- Run Controller with:
  - »python pox.py forwarding.l2_learning openflow.spanning_tree -no-flood -hold-down log.level -DEBUG samples.pretty_log openflow.discovery host_tracker info.packet_dump
- Run topology from command line with remote controller:
  - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633

POX controller with spanning tree in DEBUG mode

- Run Controller with:
    - »python pox.py forwarding.l2_learning openflow.spanning_tree -no-flood -hold-down log.level -DEBUG samples.pretty_log openflow.discovery host_tracker info.packet_dump
- Run topology from command line with remote controller:
    - sudo mn - - topo=linear,5 - - controller=remote, ip=127.0.0.1, port=6633

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

- Results:
  - OpenFlow switches may optionally support 802.1D Spanning Tree Protocol.
  - Those switches process all 802.1D packets locally before performing flow lookup.
    - Flow tables are filled with entries including "actions=CONTROLLER"
  - After pinging from one host to another,
    - forwarding.l2_learning module is run in controller per flow
    - it installs exact-match rules for each flow.
    - Host becomes reachable
    - Switches that do not support 802.1D spanning tree must allow the controller to specify the port status for packet flooding through the port-mod messages.

Link down/up

- Afterwards, try to link *down* in running topology:
  - Down the link between switch1 and switch2 by following command:
  - »link s1 s2 down
  - »h1 ping h2 -c 3 results:
    - In controller side: openflow.spanning_tree tries to find alternative link between h1 and h2. However, openflow.discovery→timeout
    - In mininet side: Destination Host Unreachable

Link down/up

- Afterwards, try to link *down* in running topology:
  - Down the link between switch1 and switch2 by following command:
  - »link s1 s2 down
  - »h1 ping h2 -c 3 results:
    - In controller side: openflow.spanning_tree tries to find alternative link between h1 and h2. However, openflow.discovery→timeout
    - In mininet side: Destination Host Unreachable

Link down/up

- Afterwards, try to link *down* in running topology:
  - Down the link between switch1 and switch2 by following command:
  - »link s1 s2 down
  - »h1 ping h2 -c 3 results:
    - In controller side: openflow.spanning_tree tries to find alternative link between h1 and h2. However, openflow.discovery→timeout
    - In mininet side: Destination Host Unreachable

Link down/up

- Afterwards, try to link *down* in running topology:
  - Down the link between switch1 and switch2 by following command:
  - »link s1 s2 down
  - »h1 ping h2 -c 3 results:
    - In controller side: openflow.spanning_tree tries to find alternative link between h1 and h2. However, openflow.discovery→timeout
    - In mininet side: Destination Host Unreachable

Link down/up

- Afterwards, try to link *down* in running topology:
  - Down the link between switch1 and switch2 by following command:
  - »link s1 s2 down
  - »h1 ping h2 -c 3 results:
    - In controller side: openflow.spanning_tree tries to find alternative link between h1 and h2. However, openflow.discovery→timeout
    - In mininet side: Destination Host Unreachable

Link down/up

- Afterwards, try to link *down* in running topology:
  - Down the link between switch1 and switch2 by following command:
  - »link s1 s2 down
  - »h1 ping h2 -c 3 results:
    - In controller side: openflow.spanning_tree tries to find alternative link between h1 and h2. However, openflow.discovery→timeout
    - In mininet side: Destination Host Unreachable

Outline

Custom Controller and Topology written in python language

DEMO!!!

Thank you for your attention.

Any questions?