Kyushu Institute of Technology

Tsuru Laboratory
Network Management Research Laboratory

Chapter 5:
MININET ADVANCED

Dr. Nguyen Viet Ha [(1)], **and Prof. Masato Tsuru** [(2)]

[(1)] **nguyen.viet-ha503@mail.kyutech.jp,** [(2)] **tsuru@cse.kyutech.ac.jp**

Kyushu Institute of Technology, Japan

November 12th, 2020

Index

1. **Using NetworkX for routing decision.**
2. **Using Scapy for create and send packet**
3. **Dynamic network topology**

1. USING NETWORKX FOR ROUTING DECISION [1]

1. Using NetworkX for routing decision

- NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

- Using NetworkX for Routing Decision (e.g., **Shortest Path First**).

1. Using NetworkX for routing decision

- Install NetworkX:

```
mininet@mininet:~$ pip install networkx
```

- To install networkx and **all optional** packages:

```
mininet@mininet:~$ pip install network[all]
```

- Upgrade NetworkX:

```
mininet@mininet:~$ pip install --upgrade networkx
```

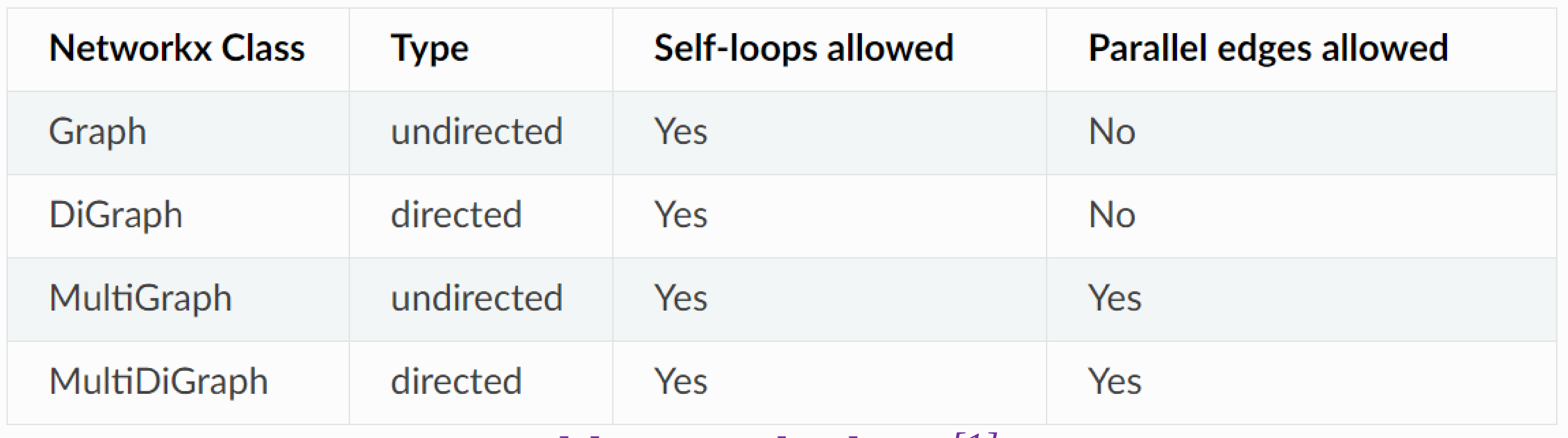
1. Using NetworkX for routing decision

- Import NetworkX to your source code:

```
import networkx as nx
```

- Declare the **Topology** (graph):

```
def __init__(self, *args, **kwargs):
    self.G = nx.Graph()
```

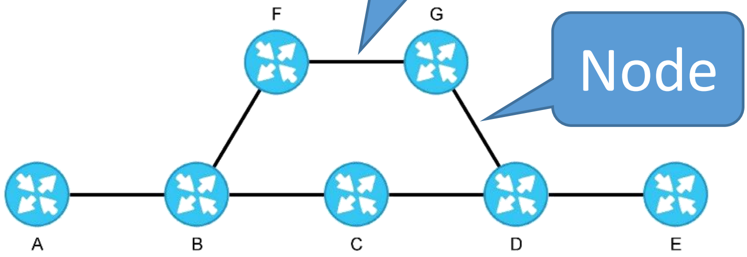


Networkx Class	Type	Self-loops allowed	Parallel edges allowed
Graph	undirected	Yes	No
DiGraph	directed	Yes	No
MultiGraph	undirected	Yes	Yes
MultiDiGraph	directed	Yes	Yes

Table: graph class [1]

1. Using NetworkX for routing dec

Edge



Node

- Add **node** to the “graph”:

```
self.G.add_node(<node>, **attr)
```

Nodes can be, for example, strings or numbers.

- Add **edge** to the “graph”:

```
self.G.add_edge(<node1>, <node2>, **attr)
```

Edge data (or labels or objects) can be assigned using keyword arguments.

- Remove **node** from the “graph”:

```
self.G.remove_node(<node)
```

- Remove **edge** to the “graph”:

```
self.G.Remove_edge(<node1>, <node2>)
```

1. Using NetworkX for routing decision

- Clear the “graph”:

```
self.G.clear()
```

- Please check more command in NetworkX document:
  - E.g., add_node_from, add_edge_from, etc
  - https://networkx.org/documentation/stable/tutorial.html#

1. Using NetworkX for routing decision

- **Hands-on-Lab:**
  - Get the initial source code from your email. We will start to modified that code.

  - **Purpose:**
    - Create the topology Graph().
    - Dynamically update the topology when network status changes.
    - Find the best route (i.e., Shortest Path First algorithm) between End-Nodes.

1. Using NetworkX for routing decision

- Create the topology Graph().
- Dynamically update the topology when network status changes.

```
import networkx as nx

def __init__(self, *args, **kwargs):
    ...
    self.G = nx.Graph()



@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):

    ...

    # Learn source MAC address and port
    if smac not in self.G:
        print("   - Add MAC address ({}) to the Database".format(smac))
        self.G.add_node(smac)
        self.G.add_edge(dpid,smac,port={dpid:pin})

    if dmac not in self.G:
        print("   - Do not know the Destination MAC Address ({}) - Cannot
find the route".format(dmac))
        return
```

Dictionary (port):

datapath ID of OFS	Port of OFS

1. Using NetworkX for routing decision

- Create the topology Graph().
- Dynamically update the topology when network status changes.

```
def BuildTopology(self):
    ...
    self.G.clear()

    for l in self.topo_raw_links:
        ...

        self.G.add_node(_dpid_src)
        self.G.add_node(_dpid_dst)
        self.G.add_edge(_dpid_src,_dpid_dst,port={_dpid_src:_port_src,
_dpid_dst:_port_dst})
```

Dictionary (port):

datapath ID of source OFS	Port of source OFS
datapath ID of destination OFS	Port of destination OFS

1. Using NetworkX for routing decision

- Find the best route (i.e., Shortest Path First algorithm) between End-Nodes.

```
@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def packet_in_handler(self, ev):
    ...

    print("\n   - DATA info: packet from {} to {}".format(smac,dmac))


    path_route = nx.shortest_path(self.G,smac,dmac)
    print ("      + Path Route: {}".format(path_route))
    for i in [1,len(path_route)-2]:
        _from = path_route[i]
        _to   = path_route[i+1]
        _exIf = self.G[_from][_to]['port'][_from]
        print("         * From {} to {} - Exit interface:
{}".format(_from,_to,_exIf))
```

1. Using NetworkX for routing decision

- **Run app:**
  - Open the **first** SSH terminal:
    - *Type:* ***sudo mn --topo linear,3,1 --switch ovsk --controller remote --arp --mac***

Populate static ARP entries of each host in each other

Auto set the simple MAC addresses

  - Open the **second** SSH terminal:
    - *Type:* ***sudo ryu-manager ~/MyApp/*** **NetworkX.py --observe-links**

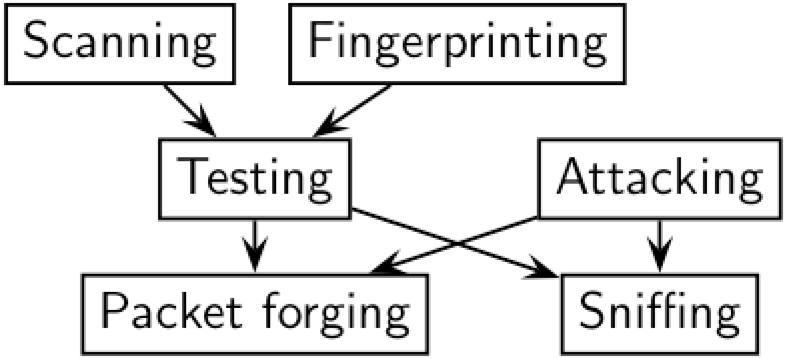
1. Using NetworkX for routing decision

- **Run app:**
  - In the **first** SSH terminal:
    - Type:
      - “h1 ping h3”
      - “h3 ping h1”
  - Check the output on the second SSH terminal

2. USING SCAPY FOR CREATE AND SEND PACKET [2]

2. Using Scapy for create and send packet

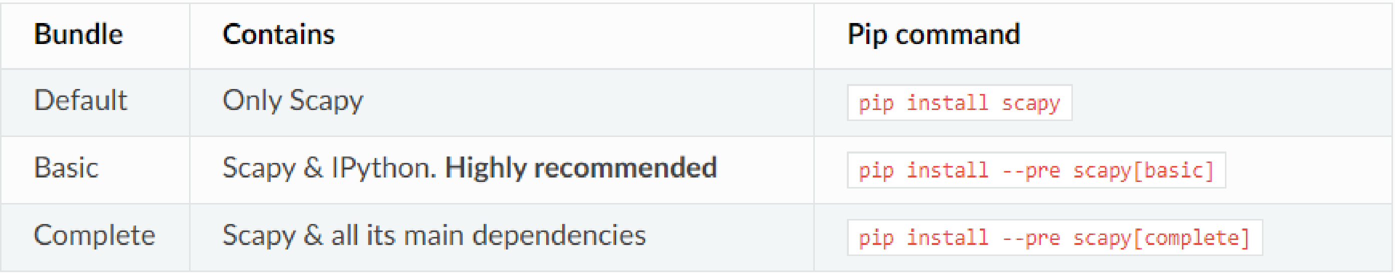
- Scapy is a Python program that enables the user to **send**, **sniff** and **dissect and forge** network packets.
  - This capability allows construction of tools that can **probe**, **scan** or **attack** networks.



2. Using Scapy for create and send packet

❖ Install Scapy

```
mininet@mininet:~$ pip install --pre scapy[basic]
```

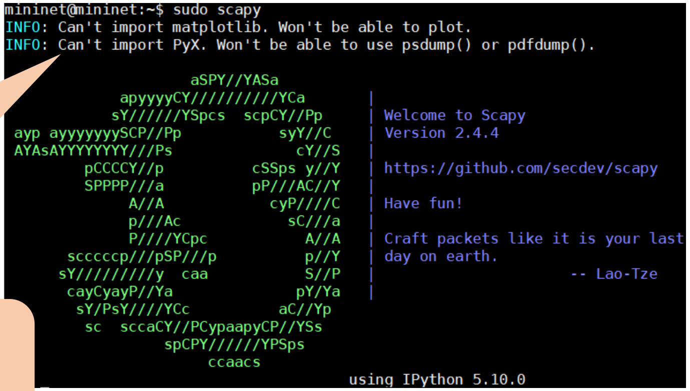


Bundle	Contains	Pip command
Default	Only Scapy	`pip install scapy`
Basic	Scapy & IPython. **Highly recommended**	`pip install --pre scapy[basic]`
Complete	Scapy & all its main dependencies	`pip install --pre scapy[complete]`

Table: Scapy comes in 3 bundles [2]

❖ Run Scapy:

```
mininet@mininet:~$ sudo scapy
```



```
mininet@mininet:~$ sudo scapy
INFO: Can't import matplotlib. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.4
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft packets like it is your last
       scccccp///pSP///p          p//Y   | day on earth.
      sY/////////y  caa           S//P   |                      -- Lao-Tze
       cayCyayP//Ya              pY/Ya   |
        sY/PsY////YCc          aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
                                       using IPython 5.10.0
```

If you do not have all optional packages installed, Scapy will inform you that some features will not be available.
The basic features of **sending and receiving packets should still work**, though.

2. Using Scapy for create and send packet

Create a packet:

```
p = lower_layer([paramenter])/…/higher_layer([paramenter])
```

- Refer the layers information is here:
  - https://scapy.readthedocs.io/en/latest/api/scapy.layers.html

- **Ex:**

```
>>>from scapy.layers.http import HTTP, HTTPRequest

>>>p = Ether(dst="00:00:00:00:00:02")/IP(dst="10.0.0.2")/
TCP(dport=80)/HTTP()/HTTPRequest(Method="GET",
Http_Version="HTTP/1.1", Path="/default/")
```

2. Using Scapy for create and send packet

Send packet:

```
>>>sendp(p, [arguments ])
```

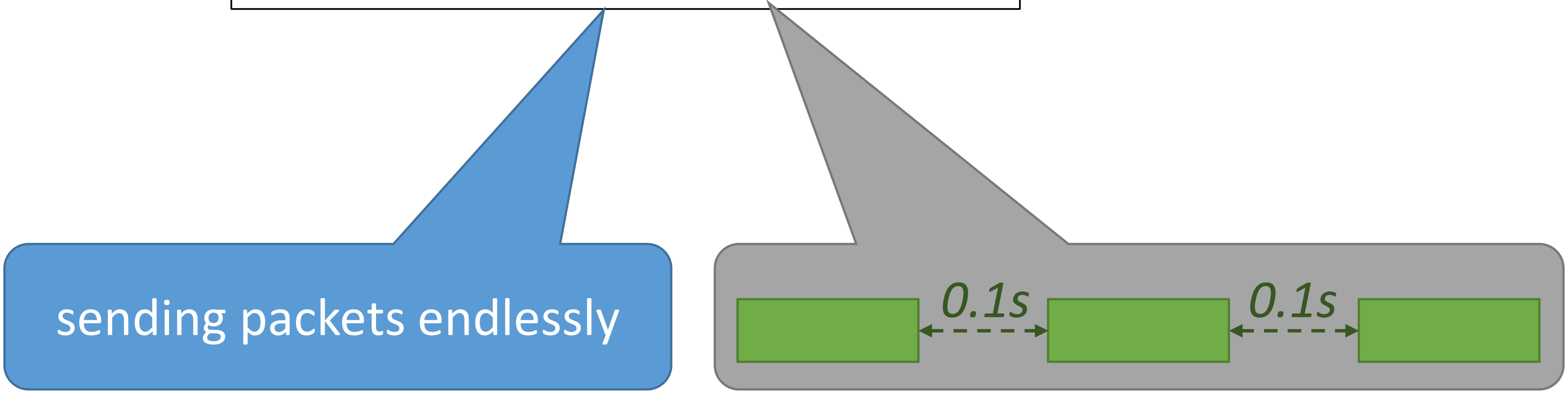
- The **send()** function will send packets at layer 3.
  - That is to say, it will handle routing and layer 2 for you.
- The **sendp()** function will work at layer 2.

- The main arguments for the send commands are:
  - **iface**: The interface to send packets.
  - **inter**: The time, in seconds, that we want to pass between package and package sent.
  - **loop**: To keep sending packets endlessly, set this to 1.
    - Stop it by pressing **Ctrl + C**.

2. Using Scapy for create and send packet

- **Send packet:**
  - Ex:

```
>>>sendp(p, loop=1, inter=0.1)
```



sending packets endlessly

2. Using Scapy for create and send packet

- **Send and receive packets:**
  - ***sr():*** sending packets and receiving answers.
    - The packets **must be layer 3 packets** (IP, ARP, etc.)
  - ***sr1():*** returns **one packet** that answered the packet (or the packet set) sent.
  - ***srp():*** do the same for **layer 2 packets.**

  - Ex:

```
>>>sr1(IP(dst="10.0.0.2")/ICMP())
```

2. Using Scapy for create and send packet

Sniffing:

- *Hint: run the lsc() to see a list of what commands Scapy has available.*
  - tcpdump(), tshark(), sniff()

- ***sniff():*** *https://scapy.readthedocs.io/en/latest/usage.html#starting-scapy → Sniffing*

```
>>> sniff(iface="h2-eth0", prn=lambda x: x.summary())
```

```
>>> sniff(iface="h2-eth0", prn=lambda x: x.show())
```

```
>>> sniff(iface="h2-eth0", filter="icmp and host 10.0.0.2",
prn=lambda x: x.show())
```

https://biot.com/capstats/bpf.html

2. Using Scapy for create and send packet

- **Hands-on-Lab:**
  - Create the **ARP request** from h1 to h2.
    - https://scapy.readthedocs.io/en/latest/api/scapy.layers.l2.html#scapy.layers.l2.ARP
  - Create the **ICMP request** from h1 to h2.
    - https://scapy.readthedocs.io/en/latest/api/scapy.layers.inet.html#scapy.layers.inet.ICMP
  - “Send and Receive” ARP and ICMP packets.
  - Capture the packet at h2 to see the result.

2. Using Scapy for create and send packet

- **Hands-on-Lab:**
  - Open the SSH terminal:
    - *Type: **sudo mn --switch ovsk --controller remote --mac***
  - Capture the packet at h2 to see the result.

```
>>> sniff(iface="h2-eth0", prn=lambda x: x.summary())
```

conf.iface

2. Using Scapy for create and send packet

- **Hands-on-Lab:**
  - Create and Send the **ARP request** from h1 to h2.
    - https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml

```
>>>arpp = Ether(dst="                    ")/ARP(hwtype=  , op=  ,
hwsrc="                        ", psrc="           ", pdst="          ")

>>>srp(arpp)
```

2. Using Scapy for create and send packet

- **Hands-on-Lab:**
  - Create and Send the **ICMP request** from h1 to h2.
    - https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml

```
>>>icmpp = IP(dst="              ")/ICMP(type=  )

>>>sr1(icmpp)
```

3. DYNAMIC NETWORK TOPOLOGY

3. Dynamic network topology

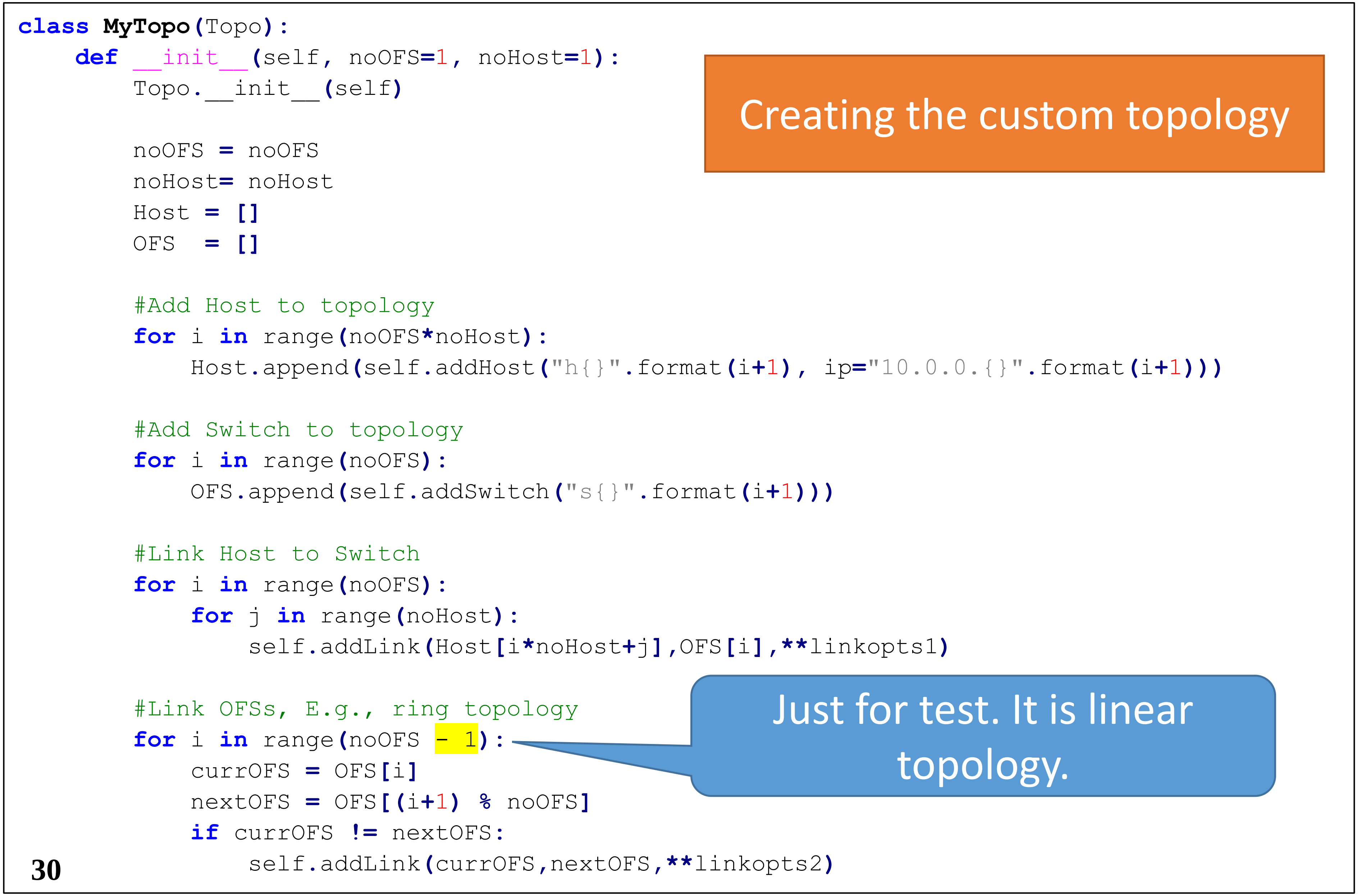
- **Run the simulation on another way:**
  - Previously, we run **"sudo mn …"** then do some tasks (e.g., ping, etc.)
  - Now, we **run the preprepare tasks with one time** (one command line).
  - Please get the file of "**customtopo.py**" from your email and study the code together.

3. Dynamic network topology

```
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController, OVSKernelSwitch
from mininet.link import TCLink, TCIntf
from mininet.util import dumpNodeConnections,dumpNetConnections, dumpPorts
from mininet.cli import CLI
from mininet.log import setLogLevel
from ryu.lib import hub

from sys import argv
import argparse

#Link parameter
linkopts1 = dict(bw=100,  delay='1ms') #Host link
linkopts2 = dict(bw=1000, delay='3ms') #Switch Link
linkopts3 = dict(loss=10, bw=1000, delay='3ms') #Switch Link, link loss=10%
```

Just import some classes and define some initial variables

3. Dynamic network topology



```
class MyTopo(Topo):
    def __init__(self, noOFS=1, noHost=1):
        Topo.__init__(self)

        noOFS = noOFS
        noHost= noHost
        Host = []
        OFS  = []

        #Add Host to topology
        for i in range(noOFS*noHost):
            Host.append(self.addHost("h{}".format(i+1), ip="10.0.0.{}".format(i+1)))

        #Add Switch to topology
        for i in range(noOFS):
            OFS.append(self.addSwitch("s{}".format(i+1)))

        #Link Host to Switch
        for i in range(noOFS):
            for j in range(noHost):
                self.addLink(Host[i*noHost+j],OFS[i],**linkopts1)

        #Link OFSs, E.g., ring topology
        for i in range(noOFS - 1):
            currOFS = OFS[i]
            nextOFS = OFS[(i+1) % noOFS]
            if currOFS != nextOFS:
                self.addLink(currOFS,nextOFS,**linkopts2)
```

Creating the custom topology

Just for test. It is linear topology.

3. Dynamic network topology

```
def main(*args):
    parser = argparse.ArgumentParser()
    parser.add_argument('--NumOFS',  type=int, action="store", default=5)
    parser.add_argument('--NumHost', type=int, action="store", default=1)
    args = parser.parse_args()

    NumOFS  = args.NumOFS
    NumHost= args.NumHost

    mytopo = MyTopo(NumOFS,NumHost)
    net  = Mininet(topo=mytopo, switch=OVSKernelSwitch, controller=RemoteController("c0",
                   ip="127.0.0.1"), autoSetMacs=True, link=TCLink)

    #Run default command from hosts. E.g., Disable IPv6:
    for h in net.hosts:
        h.cmd("sysctl -w net.ipv6.conf.all.disable_ipv6=1")
        h.cmd("sysctl -w net.ipv6.conf.default.disable_ipv6=1")

    #Start simulation --------------------------
    net.start()

    #Stop simulation ----------------------------
    net.stop()
```

Pass some variables from the command line

Create the topology

Add some tasks here

```
if __name__ == "__main__":
    setLogLevel("info")
    main()
```

3. Dynamic network topology

- **Run the simulation on another way:**
  - Run simulation:
    - Open the **first** SSH terminal:
      - *Type:* ***sudo ryu-manager ~/MyApp/switch_ofp1_3.py***
    - Open the **second** SSH terminal:
      - *Type:* ***sudo python customtopo.py***
      - *And:*
      - *Type:* ***sudo python customtopo.py --numHost=1 --numOFS=5***

3. Dynamic network topology

- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 1:***

```
#Start simulation --------------------------
net.start()

#Access CLI console--------------------------
CLI(net)

#Stop simulation ----------------------------
net.stop()
```

3. Dynamic network topology

- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 2:***

```
#Start simulation --------------------------
net.start()

#Show topology infomation-------------------
#List host
print("\nHost list:")
dumpNodeConnections(net.hosts)

#List connections
print("\nConnection/Link list:")
dumpNetConnections(net)

#List port
print("\nPort list:")
dumpPorts(net.switches)

#Stop simulation ----------------------------
net.stop()
```

3. Dynamic network topology

- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 3:***

```
#Start simulation ---------------------------
net.start()


#Ex: h1 ping h3
hub.sleep(1)
h1 = net.getNodeByName("h1")
print(h1.cmd("ping 10.0.0.3 -c 1 "))

hub.sleep(1)
h3 = net.getNodeByName("h3")
print(h3.cmd("ping 10.0.0.1 -c 10"))

#Stop simulation -----------------------------
net.stop()
```

3. Dynamic network topology

- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 4:***
      - In this example, we will test on **your submitted assignment** (e.g., Assigment2.py).
      - We will test with the loop topology (e.g., ring topology) to see the **route decision change** when the **network changes**.
        - Please make sure your ARP handling is work correctly.
        - **No any BROADCAST packet** can forward to the network. If not, the **Broadcast Storm will happen** on you network.

3. Dynamic network topology

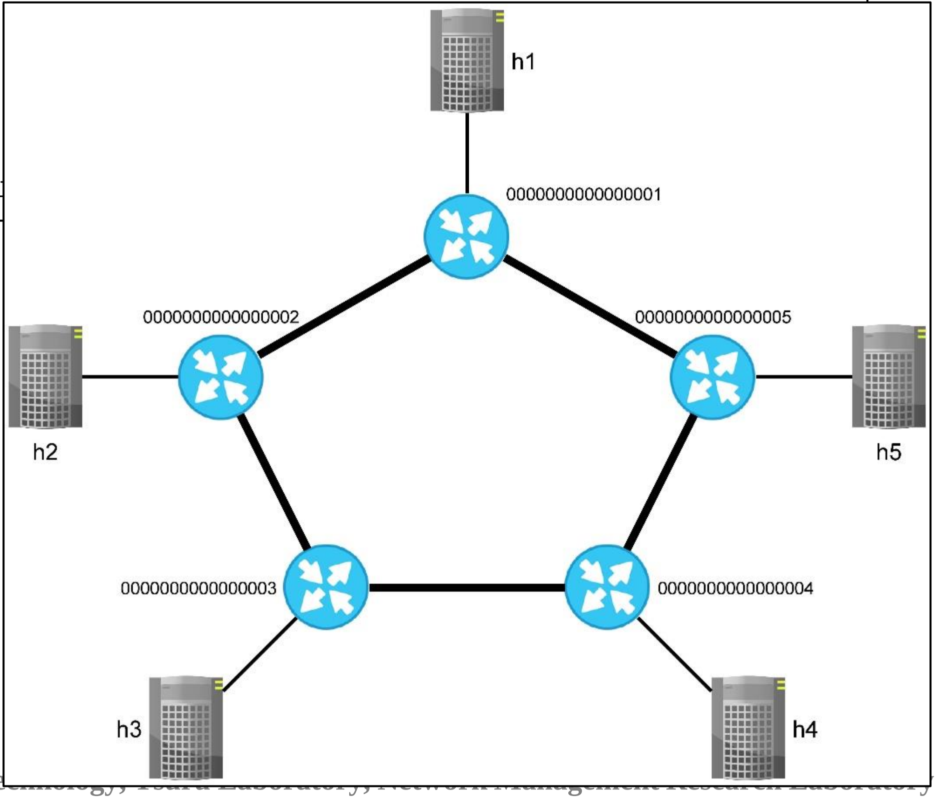
- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 4:***
      - Open the **first** SSH terminal:
        - *Type:* ***sudo ryu-manager ~/MyApp/*****Assigment2.py --observe-links**

3. Dynamic network topology

- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 4:***

```
#Link OFSs, E.g., ring topology
for i in range(noOFS - 1):
    currOFS = OFS[i]
    nextOFS = OFS[(i+1) % noOFS]
    if currOFS != nextOFS:
       self.addLink(currOFS,nextOFS,**l
```

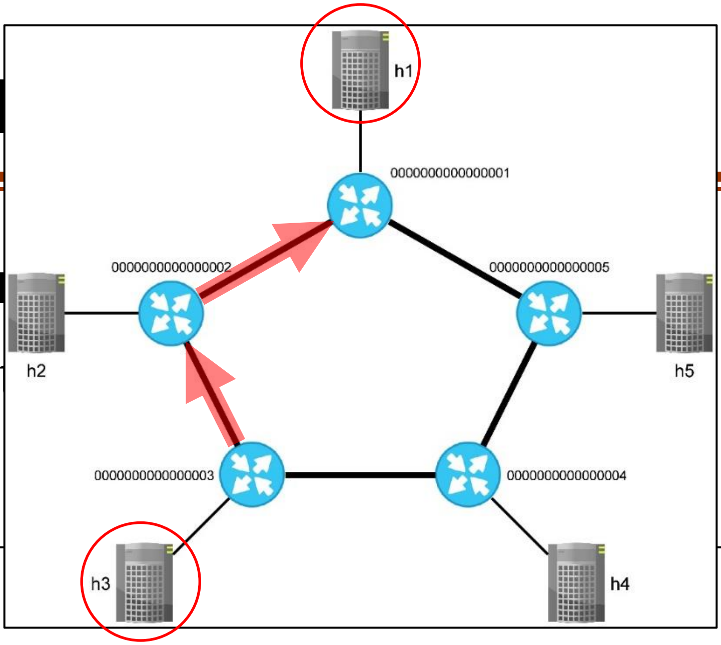
Change to:
for i in range(**noOFS**)

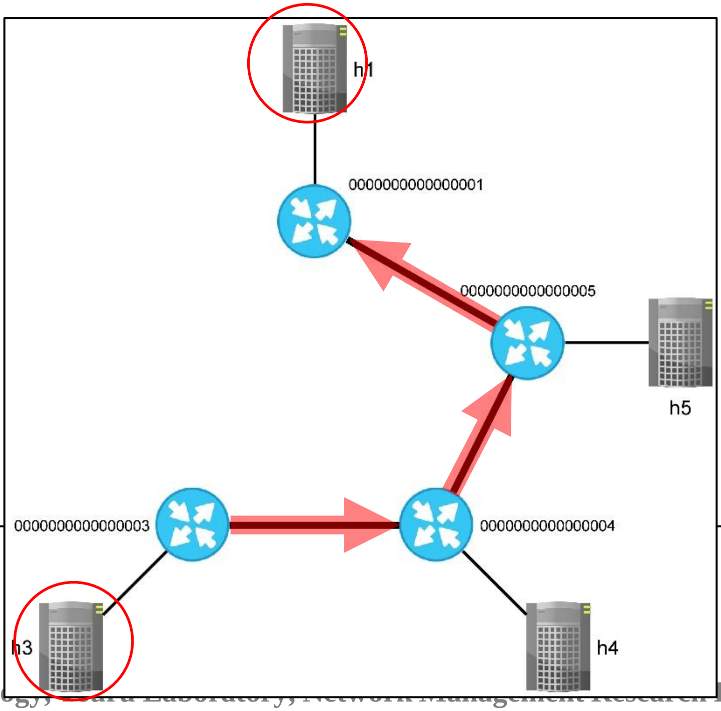


3. Dynamic network top

- **Run the simulation on another**
  - Add some tasks to the simulat
    - ***Example 4:***

```
#Start simulation ---------------------------
net.start()

#Ex: h1 ping h3
...

hub.sleep(1)
s2 = net.getNodeByName("s2")
s2.stop()

hub.sleep(5)
print(h3.cmd("ping 10.0.0.1 -c 1"))


#Stop simulation ----------------------------
net.stop()
```





`s2.start(net.controllers)`

3. Dynamic network topology

- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 4:***
      - ***FYI: add new nodes:***

```
nh1 = net.addHost ("nh1")
ns1  = net.addSwitch("ns1",dpid="0000000000000010",cls=OVSKernelSwitch)
net.addLink  (nh1,ns1,**linkopts1)
nh1.setIP("10.0.0.3")

s1 = net.getNodeByName("s1")
slink = net.addLink  (ns1 ,s1,**linkopts1)

s1.attach(slink.intf2)
ns1.start(net.controllers)
```

3. Dynamic network topology

- **Run the simulation on another way:**
  - Add some tasks to the simulation:
    - ***Example 5:***
      - In this example, we will change the link parameters
        - Ex: link loss rate

3. Dynamic network topology

```
#Start simulation ---------------------------
net.start()


hub.sleep(1)
h1 = net.getNodeByName("h1")
print(h1.cmd("ping 10.0.0.2 -c 1"))

hub.sleep(1)
h2 = net.getNodeByName("h2")
print(h2.cmd("ping 10.0.0.1 -c 10"))


s1 = net.getNodeByName("s1")
s2 = net.getNodeByName("s2")
connections = s1.connectionsTo(s2)

(intf, link_intf) = connections[0]
intf.config(**linkopts3)

hub.sleep(1)
print(h2.cmd("ping 10.0.0.1 -c


#Stop simulation ------------        ------
net.stop()
```

Get links (connections) between s1 and s2

Get interface of s1 attached on the link

Change the interface parameters

*** If change BW delay, we should change on both interfaces

```
linkopts3 = dict(loss=10, bw=1000, delay='3ms')
```

Reference

[1] “Software for Complex Networks,” [Online], available https://networkx.org/documen-tation/stable/index.html, accessed in Nov. 2020

[2] “Welcome to Scapy’s documentation!,” [Online], available: https://scapy.readthedocs.io/en/latest/, accessed in Nov. 2020

Kyushu Institute of Technology

Tsuru Laboratory

Network Management Research Laboratory

Thank you for your attention

Nguyen Viet Ha

nguyen.viet-ha503@mail.kyutech.jp

Masato Tsuru

tsuru@cse.kyutech.ac.jp

Kyushu Institute of Technology, Japan