软件定义网络

实 验 报 告（一）

2025-2026 学年　第一学期

姓　　名	郭振乾
学　　号	202313407281
班　　级	网络 2302
指 导 老 师	何亨
实 验 日 期	2025.10.27

实验名称　网络模拟软件 Mininet 和软件交换机 OVS 使用

一、实验目的

能够运用 Mininet 命令行、Python 脚本等方式创建网络拓扑结构。

能够运用 OpenvSwitch 支持的 OpenFlow 命令对流表进行操作。

二、实验内容

1、运用 Mininet 命令行（使用 sudo mn 命令）创建如下网络拓扑：

（1）最小的拓扑，即一个交换机下挂两个主机；

（2）包含 4 台主机和 4 台交换机的线型拓扑；

（3）包含 3 台主机和 1 台交换机的星型拓扑；

（4）深度为 2 扇出为 2 的树型拓扑；

（5）自定义拓扑，如图 1：

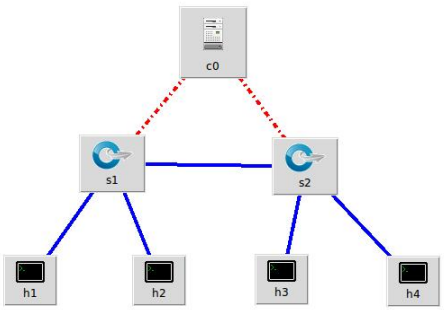


图 1　自定义拓扑 1

2、运用 Python 脚本（使用 sudo python 命令）创建如图 2 所示拓扑：

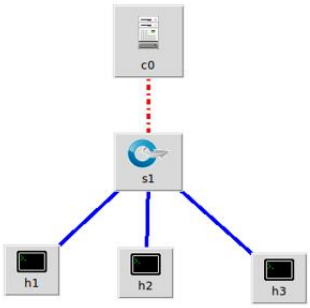


图 2　自定义拓扑 2

3、参照图 1 所示的拓扑部署一个基于 SDN 架构的网络环境，具体要求如下：

（1）采用 Mininet 模拟网络环境，连接 Floodlight 控制器。

（2）使用 OpenvSwitch 流表管理命令 ovs-ofctl 下发流表项，要求按 IP 地址匹配数据流，使得 h1 与 h3 可以 ping 通，h2 与 h4 可以 ping 通，其他主机之间都不能 ping 通。

三、实验步骤及结果

1、（1）sudomn --topominimal

```
sdn@sdn:~$ sudo mn --topo minimal
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

(2）sudomn --topo linear,4

```
sdn@sdn:~$ sudo mn --topo linear,4
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> exit
```

(3）sudomn --topo single,3

```
sdn@sdn:~$ sudo mn --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

(4）sudomn --topo tree,fanout=2,depth=2

```
sdn@sdn:~$ sudo mn --topo tree,fanout=2,depth=2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

（5）

```python
from mininet.topo import Topo

class MyTopo(Topo):
    """自定义拓扑类 - 2交换机4主机"""

    def build(self):
        """构建网络拓扑结构"""

        # 创建交换机
        s1 = self.addSwitch('s1')  # 核心交换机1
        s2 = self.addSwitch('s2')  # 核心交换机2

        # 创建主机
        h1 = self.addHost('h1')  # 主机1
        h2 = self.addHost('h2')  # 主机2
        h3 = self.addHost('h3')  # 主机3
        h4 = self.addHost('h4')  # 主机4

        # 建立连接
        self.addLink(s1, h1)  # s1连接h1
        self.addLink(s1, h2)  # s1连接h2
        self.addLink(s2, h3)  # s2连接h3
        self.addLink(s2, h4)  # s2连接h4
        self.addLink(s1, s2)  # 交换机互联

# 注册拓扑
topos = {'mytopo': (lambda: MyTopo())}
```

```
sdn@sdn:~/python$ sudo mn --custom Topo.py --topo mytopo
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2
*** Adding links:
(s1, h1) (s1, h2) (s1, s2) (s2, h3) (s2, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
```

2,

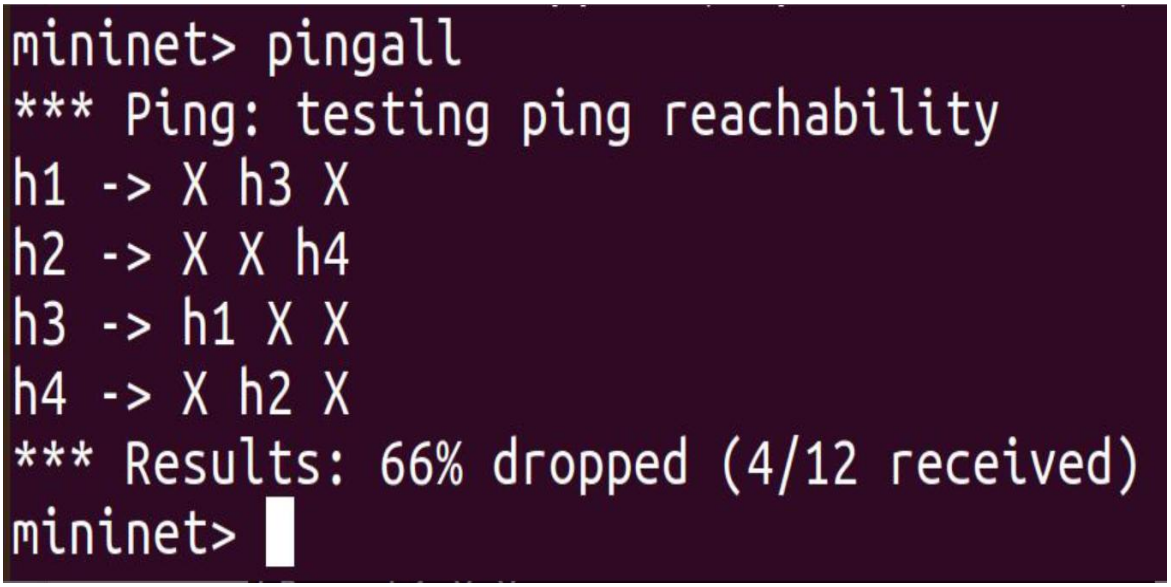
```python
from mininet.net import Mininet
from mininet.node import Controller, OVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info

def createTopology():

    net = Mininet(controller=Controller, switch=OVSSwitch)

    info("*** Adding controller\n")

    c0 = net.addController('c0')

    info("*** Adding switch\n")

    s1 = net.addSwitch('s1')
    s2 = net.addSwitch('s2')

    info("*** Adding hosts\n")

    h1 = net.addHost('h1', ip='10.0.0.1/24')
    h2 = net.addHost('h2', ip='10.0.0.2/24')
    h3 = net.addHost('h3', ip='10.0.0.3/24')
    h4 = net.addHost('h4', ip='10.0.0.4/24')

    info("*** Creating links\n")

    net.addLink(h1, s1)
    net.addLink(h2, s1)
    net.addLink(s1, s2)
    net.addLink(s2, h3)
    net.addLink(s2, h4)

    info("*** Starting network\n")
    net.start()

    info("*** Testing network connectivity\n")
    net.pingAll()

    info("*** Running CLI (type 'exit' to stop)\n")
    CLI(net)

    info("*** Stopping network\n")
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    createTopology()
```

sudopythonTopox.py

```
*** Starting network
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Testing network connectivity
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Running CLI (type 'exit' to stop)
*** Starting CLI:
mininet> pingall
```

3,

```
sdn@sdn:~$ sudo ovs-ofctl add-flow s1 priority=30,dl_type=0x0800,nw_src=10.0.0.1,nw_dst=10.0.0.2,actio
ns=drop
[sudo] password for sdn:
sdn@sdn:~$ sudo ovs-ofctl add-flow s1 priority=30,dl_type=0x0800,nw_src=10.0.0.1,nw_dst=10.0.0.2,actio
ns=drop
sdn@sdn:~$ sudo ovs-ofctl add-flow s1 priority=30,dl_type=0x0800,nw_src=10.0.0.1,nw_dst=10.0.0.4,actio
ns=drop
sdn@sdn:~$ sudo ovs-ofctl add-flow s1 priority=30,dl_type=0x0800,nw_src=10.0.0.2,nw_dst=10.0.0.3,actio
ns=drop
sdn@sdn:~$ sudo ovs-ofctl add-flow s1 priority=30,dl_type=0x0800,nw_src=10.0.0.2,nw_dst=10.0.0.3,actio
ns=drop
sdn@sdn:~$ sudo ovs-ofctl add-flow s2 priority=30,dl_type=0x0800,nw_src=10.0.0.3,nw_dst=10.0.0.4,actio
ns=drop
```



四、实验小结

本次实验聚焦 SDN 核心工具与实操应用，让我系统梳理了 Mininet 拓扑构建与 Open vSwitch 流表配置的关键知识点。实验中，我不仅掌握了命令行快速搭建基础拓扑、Python 脚本自定义复杂网络拓扑的两种核心方法，还深入理解了控制器、交换机、主机之间的通信逻辑，以及 SDN 控制平面与数据平面分离的架构优势。

在流表配置与流量控制环节，我曾遭遇不同网段主机无法通信的问题。通过逐一排查 ovs-ofctl show 设备状态、dump-flows 流表规则，发现是流表匹配字段中 IP 网段掩码设置错误，且未添加跨网段转发规则。后续通过修正掩码参数、补充转发流表，并利用 ping 命令逐步验证，最终实现了指定网段互通、不同网段隔离的预期效果。此外，在重复实验时，我还遇到了旧流表残留导致的配置冲突，通过 ovs-ofctl del-flows 清空规则后，实验得以顺利推进。

此次实验不仅深化了我对 SDN 技术原理的理解，更锻炼了我排查网络故障的逻辑思维与实操能力，让我意识到实验操作中 “及时清理环境、细致核对参数” 的重要性，为后续学习 SDN 控制器开发、网络虚拟化等进阶内容积累了宝贵的实践经验。