

# mininet介绍

步骤：

安装虚拟机；

[【Ubuntu Linux20.04入门】Win7系统下，下载与安装虚拟机VirtualBox、安装Ubuntu20.04超详细图文步骤\\_win7 安装virtualbox-CSDN博客](#)

联网后，下载mininet软件包；

[Ubuntu20.04中搭建SDN环境（超详细）\\_sdn环境搭建-CSDN博客](#)

[Ubuntu下mininet的安装方法\\_mininet安装-CSDN博客](#)

下载源码：

```
git clone git://github.com/mininet/mininet
```

然后进入mininet/util目录

```
cd mininet  
cd util  
sudo ./install.sh -n3v
```

## 1. Mininet的概念

Mininet是由斯坦福大学基于Linux Container架构开发的一个进程虚拟化网络仿真工具，可以创建一个包含主机，交换机，控制器和链路的虚拟网络，其交换机支持OpenFlow，具备高度灵活的自定义软件定义网络。

## 2. Mininet的用处

为OpenFlow应用程序提供一个简单，便宜的网络测试平台； 启用复杂的拓扑测试，无需连接物理网络； 具备拓扑感知和OpenFlow感知的CLI，用于调试或运行网络范围的测试； 支持任意自定义拓扑，主机数可达4096，并包括一组基本的参数化拓扑； 提供用户网络创建和实验的可拓展Python API。



# Mininet文件结构



## 运行文件

bin/mn：主运行文件，应用Python程序编写，定义了MininetRunner类，执行sudo mn即调用本程序，是模拟网络的主程序，为整个测试创建基础平台。

## 核心代码

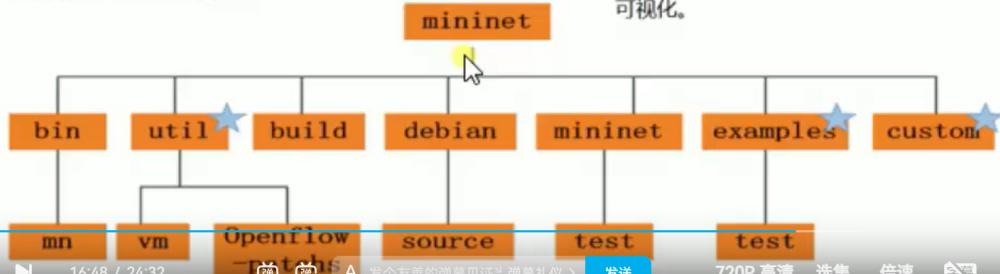
核心代码基本都在mininet子目录。

## 其他文件

custom/：放置自定义的Python文件，用来自定义拓扑。

util/：放置辅助文件，包括安装脚本、文档辅助生成等。

examples/：含有很多使用案例，包括miniedit可视化。



## 3. Mininet的优势

Mininet结合了许多仿真器，硬件测试床和模拟器的优点：

- 与仿真器比较：启动速度快；拓展性大；带宽提供多；方便安装，易使用。
- 与模拟器比较：可运行真实的代码；容易连接真实的网络。
- 与硬件测试床比较：便宜；快速重新配置及重新启动。

## 4. Mininet的主要特性

Mininet作为一个轻量级软定义网络研发和测试平台，其主要特性包括：

支持OpenFlow、Open vSwitch等软定义网络部件；方便多人协同开发；支持系统级的还原测试；支持复杂拓扑、自定义拓扑；提供python API；很好的硬件移植性（Linux兼容），结果有更好的说服力；高扩展性，支持超过4096台主机的网络结构。

## 2. Mininet常用命令

### 2.1 创建网络拓扑 常用参数

命令	含义
-clean	释放之前创建拓扑时占用的未释放的资源
-h	查看帮助
-custom	用于创建自定义拓扑
-topo	在mininet启动时通过命令行定义拓扑
-switch	定义要使用的交换机， 默认使用oVSK交换机
-mac	自动设置设备的MAC地址从而使MAC地址更易读
-controller	定义要使用的控制器，如果没有指定则使用mininet中默认的控制器(可选的有default、 remote等)
mn	创建默认最小拓扑

### 2.2 常用的内部交换命令

命令	含义
mininet >help	获取帮助列表
mininet >nodes	查看网络拓扑中结点的状态
mininet >links	显示链路健壮性信息
mininet >net	显示网络拓扑
mininet >dump	显示每个节点的接口设置和表示每个节点的进程的PID
mininet > pingall	在网络中的所有主机之间执行ping测试
mininet > pingpair	只测试前两个主机的连通性
mininet >xterm h1	打开host 1的终端
mininet >iperf	两个节点之间进行iperf tcp带宽测试 (iperf h1 h2)
mininet >iperfudp	两节点之间进行iperf udp带宽测试 (iperfudp bw h1 h2)

mininet >link	禁用/启用节点间链路 (启用 s1 s2间的链路 link s1 s2 up; 禁用s1 s2间的链路 link s1 s2 down)
mininet >h1 ping h2	h1和h2节点之间执行 ping测试
mininet >h1ifconfig	查看host1的IP 等信息
mininet >exit / quit	退出mininet登录

## 5.Mininet的命令

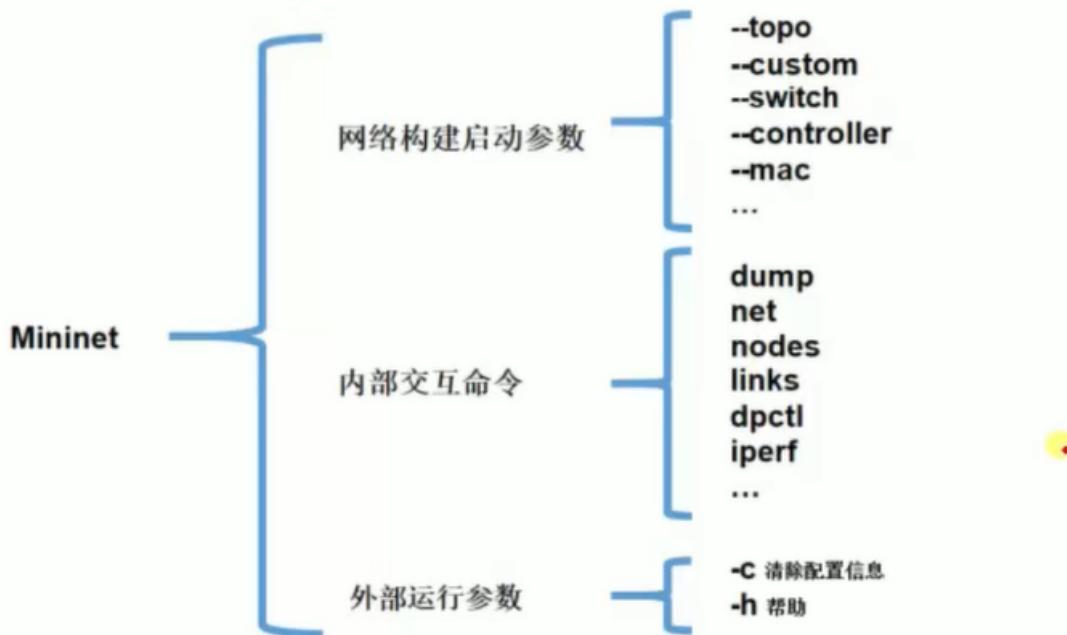
```
root@hyy-VirtualBox:/home/hyy/mininet/mininet# mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> █
```

- h 代表 host 即主机
- s 代表 switch 即交换机 (这里的交换机不是普通的交换机，是openvswitch 是一种支持 SDN(OpenFlow 协议)的交换机)
- c 代表 controller 即 控制器(SDN控制器)

清除之前的配置信息

```
mn -c
```

常用命令：



## 5.1 网络构建参数——topo

网络拓扑结构：

### (1) 单一拓扑和线性拓扑

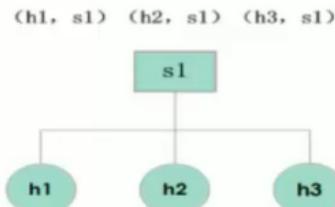
## 网络构建参数——topo (1)



### 单一 ( single ) 拓扑

整个网络拓扑中交换机有且只有一个，其可以下挂一个或多个主机

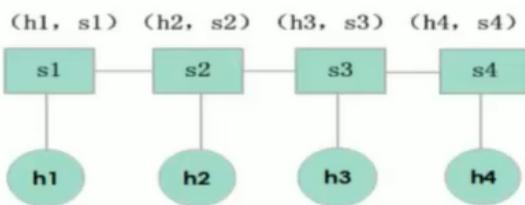
`sudo mn --topo=single,3`



### 线形 ( linear ) 拓扑

交换机连接呈线形排列，且每个交换机所连接主机数目只有一个

`sudo mn --topo=linear,4`



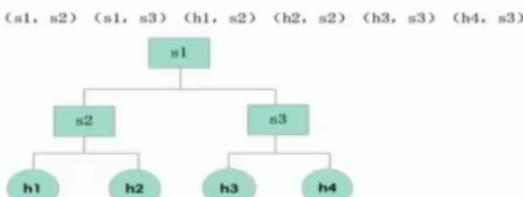
### (2) 树形拓扑和自定义拓扑

# 网络构建参数——topo (2)

## 树形 (tree) 拓扑

交换机连接成树形排列，且每个交换机所连接主机一般有多个

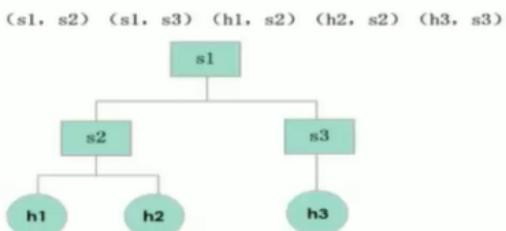
sudo mn --topo=tree,  
depth=2,fanout=2



## 自定义 (custom) 拓扑

Python编写文件file.py，执行此脚本即可  
创建定义的拓扑，--custom与--topo联用

sudo mn --custom file.py --  
topo mytopo



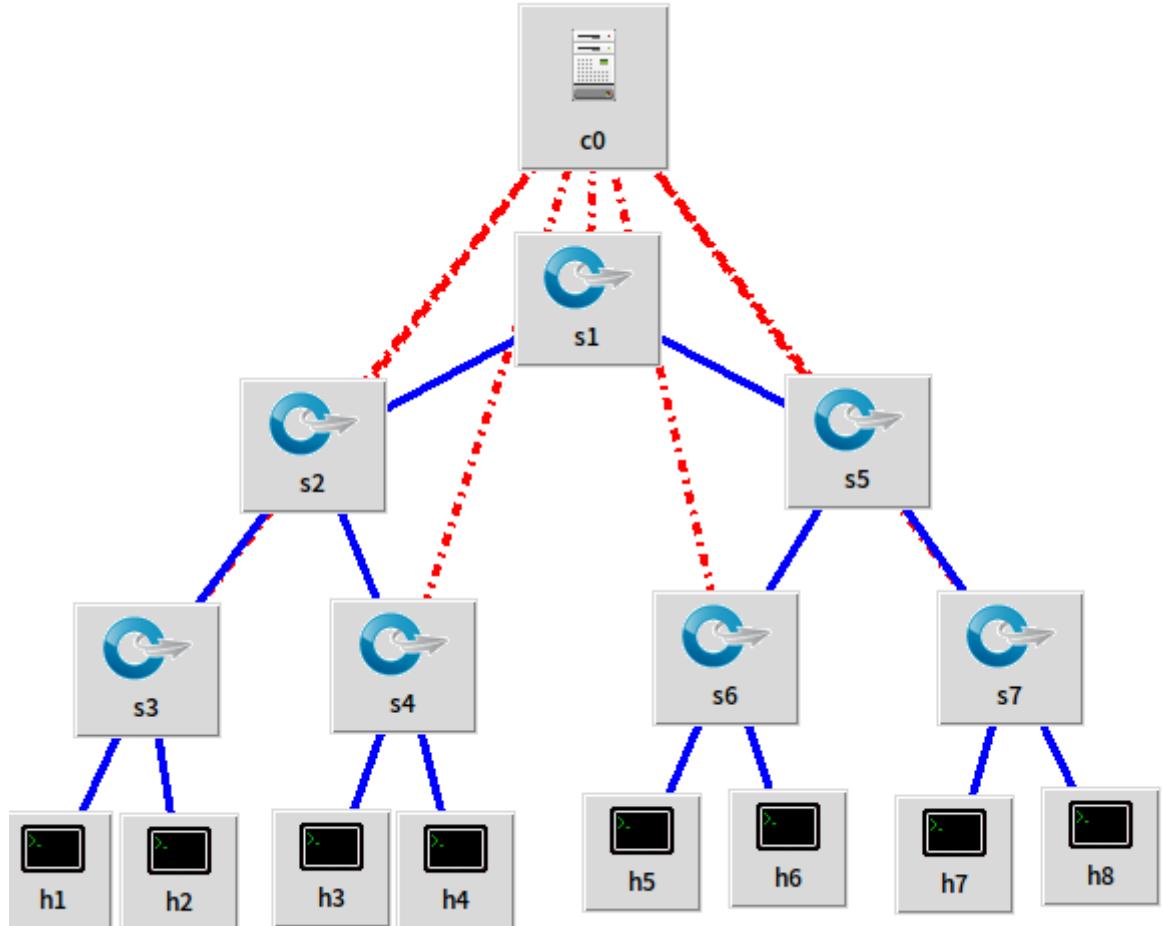
## 树形拓扑

树状结构，m层交换机，扩散为n的次方增长，终端总数为n^m台

```
mn --topo=tree,depth=3,fanout=2
#           深度为3，扇出为2
#           m           n
```

```
root@mininet-virtual-machine:~# mn --topo=tree,depth=3,fanout=2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6)
(s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet> 
```

CSDN @小白的孤独历险记



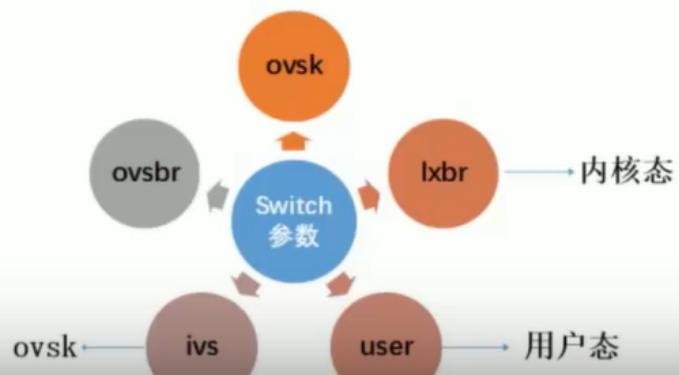
CSDN @小白的孤独历险记

## 5.2 网络构建参数——switch

### 网络构建参数——switch



**--switch** 定义mininet要使用的交换机（默认使用OVSK，即OpenVSwitch交换机）



用户态：进程可被抢占；

内核态：进程不可被抢占；

### 5.3 网络构建参数——controller

远程登陆ip地址和端口；

## 网络构建参数——controller



**--controller** 定义要使用的控制器，如果没有指定则使用mininet中默认的控制器

连接远程控制器，可以指定存在于本机或者与之相连通设备上的控制器，指定远程控制器方法：

```
sudo mn --controller=remote,--ip=[controller IP],--port=[port]
```

### 5.4 网络构建参数——mac

## 网络构建参数——mac



**--mac** 自动设置设备的MAC地址

让MAC地址易读，即 设置交换机的MAC、主机MAC及IP地址从小到大排序，且设置简单唯一，不仅让机器容易获取，也容易让肉眼很容易识别其ID。使用方法：

```
sudo mn --topo=tree,depth=2,fanout=2,--mac
```

## 5.5 内部交互命令

### 内部交互命令



使用mn默认创建网络，使用一系列命令查看并验证网络系统的链路健壮性。

```
mininet> net
```

```
mininet> nodes  
available nodes are:  
h1 h2 s1
```

```
h1 h1-eth0:s1-eth1  
h2 h2-eth0:s1-eth2  
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
```

```
mininet> links
```

```
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> h2  
h2 -> h1  
*** Results: 0% dropped (2/2 received)
```

```
h1-eth0<->s1-eth1 (OK OK)  
h2-eth0<->s1-eth2 (OK OK)
```



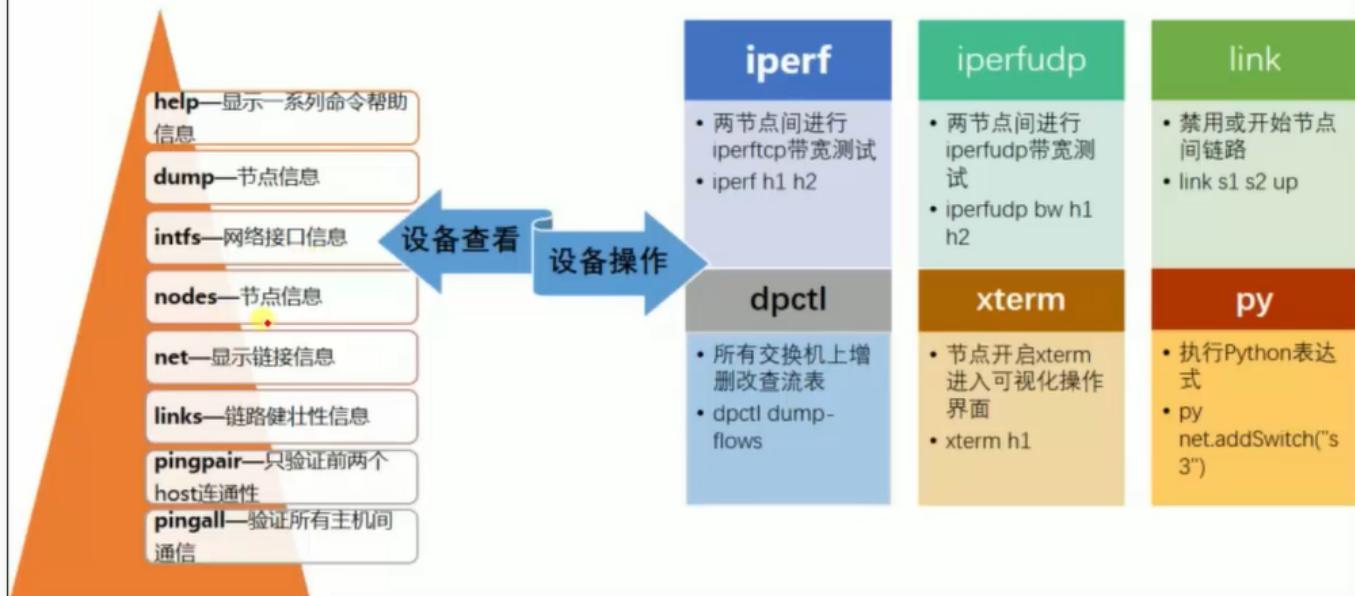
### 使用mn创建的默认网络

```
root@hyv-VirtualBox:/home/hyy/mininet/mininet# mn  
*** 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>
```

```
mininet>  
mininet> net  
h1 h1-eth0:s1-eth1  
h2 h2-eth0:s1-eth2  
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0  
c0  
mininet> nodes  
available nodes are:  
c0 h1 h2 s1  
mininet> links  
h1-eth0<->s1-eth1 (OK OK)  
h2-eth0<->s1-eth2 (OK OK)  
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> h2  
h2 -> h1  
*** Results: 0% dropped (2/2 received)
```

## 5.6 常用内部交互命令

# 常用内部交互命令



下面简单介绍py的用法：

## (1) 添加主机

```
mininet> py net.addHost('h3')
mininet> py net.addHost('h3')
<Host h3: pid=5925>
mininet>
```

## (2)添加链路

```
mininet> py net.addLink(s1,net.get('h3'))
<mininet.link.Link object at 0x79791f9ce200>
mininet>
```

还需要添加接口

```
mininet> py s1.attach('s1-eth3')
mininet>
```

给h3添加ip地址

```
mininet> py net.get('h3').cmd('ifconfig h3-eth0 10.3')
mininet>
```

使用dump查看网络信息：

```

mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5563>
<Host h2: h2-eth0:10.0.0.2 pid=5565>
<Host h3: h3-eth0:None pid=5925>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=5570>
<OVSController c0: 127.0.0.1:6653 pid=5556>
mininet>

```

使用ping命令互通：

```

<OVSController c0: 127.0.0.1:6653 pid=5556>
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=1.07 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.221 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.047 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.050 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.075 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.047 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=0.050 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=0.064 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=0.049 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=0.051 ms
64 bytes from 10.0.0.3: icmp_seq=12 ttl=64 time=0.041 ms
64 bytes from 10.0.0.3: icmp_seq=13 ttl=64 time=0.050 ms

```

再次使用dump查看

```

[root@rhel7 ~]# ./mininet.py -c ./topo1.py > ./log1.log
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5563>
<Host h2: h2-eth0:10.0.0.2 pid=5565>
<Host h3: h3-eth0:10.0.0.3 pid=5925>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=5570>
<OVSController c0: 127.0.0.1:6653 pid=5556>
mininet>

```

### (3) py的用法

使用dir命令查看函数

```

mininet> py dir(s1)
['__IP__', '__MAC__', 'OVSVersion', 'TCReapply', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_popen', '_uuids', 'addIntf', 'argmax', 'attach', 'batch', 'batchShutdown', 'batchStartup', 'bridgeOpts', 'checkSetup', 'cleanup', 'cmd', 'cmdPrint', 'cmds', 'commands', 'config', 'configDefault', 'connected', 'connectionsTo', 'controlIntf', 'controllerUUIDs', 'datapath', 'decoder', 'defaultDpid', 'defaultIntf', 'delIntf', 'deleteIntfs', 'detach', 'dpctl', 'dpid', 'dpidLen', 'execed', 'failMode', 'fdToNode', 'inNamespace', 'inToNode', 'inband', 'intf', 'intfIsUp', 'intfList', 'intfNames', 'intfOpts', 'intfs', 'isOldOVS', 'isSetup', 'lastCmd', 'lastPid', 'linkTo', 'listenPort', 'master', 'monitor', 'mountPrivateDirs', 'name', 'nameToIntf', 'newPort', 'opts', 'outToNode', 'params', 'pexec', 'pid', 'pollOut', 'popen', 'portBase', 'ports', 'privateDirs', 'protocols', 'read', 'readbuf', 'readline', 'reconnectms', 'sendCmd', 'sendInt', 'setARP', 'setDefaultRoute', 'setHostRoute', 'setIP', 'setMAC', 'setParam', 'setup', 'shell', 'slave', 'start', 'startShell', 'stdin', 'stdout', 'stop', 'stp', 'terminate', 'unmountPrivateDirs', 'vsctl', 'waitExited', 'waitOutput', 'waitForReadable', 'waiting', 'write']
mininet>

```

使用help命令查看用法

```

mininet> py help(s1)
mininet>

```

## 6.Mininet可视化--MiniEdit

# Mininet可视化



Miniedit可视化，直接在界面上编辑任意拓扑，生成python自定义拓扑脚本，简单方便。

Mininet 2.2.0+内置miniedit。在 mininet/examples下提供miniedit.py脚本，执行脚本后显示可视化界面，可自定义拓扑及配置属性。

## Miniedit三步走



- ◆ Miniedit启动
- ◆ 自定义创建拓扑，设置设备信息
- ◆ 运行拓扑并生成拓扑脚本

### 启动miniedit可视化界面

```
root@hyy-VirtualBox:/home/hyy/mininet/mininet# cd examples/
root@hyy-VirtualBox:/home/hyy/mininet/mininet/examples# ls
baressh.py      controllers2.py  limit.py        multitest.py    scratchnetuser.py
bind.py         controllers.py   linearbandwidth.py natnet.py      simpleperf.py
clustercli.py   controlnet.py   linuxrouter.py  nat.py       sshd.py
clusterdemo.py  cpu.py        miniedit.py    numberedports.py test
clusterperf.py  emptynet.py   mobility.py    openpoll.py   tree1024.py
cluster.py      hwintf.py    multilink.py  open.py      treeping64.py
clusterSanity.py __init__.py   multiping.py  README.md   vlanhost.py
consoles.py     intfoptions.py multipoll.py scratchnet.py
root@hyy-VirtualBox:/home/hyy/mininet/mininet/examples#
```

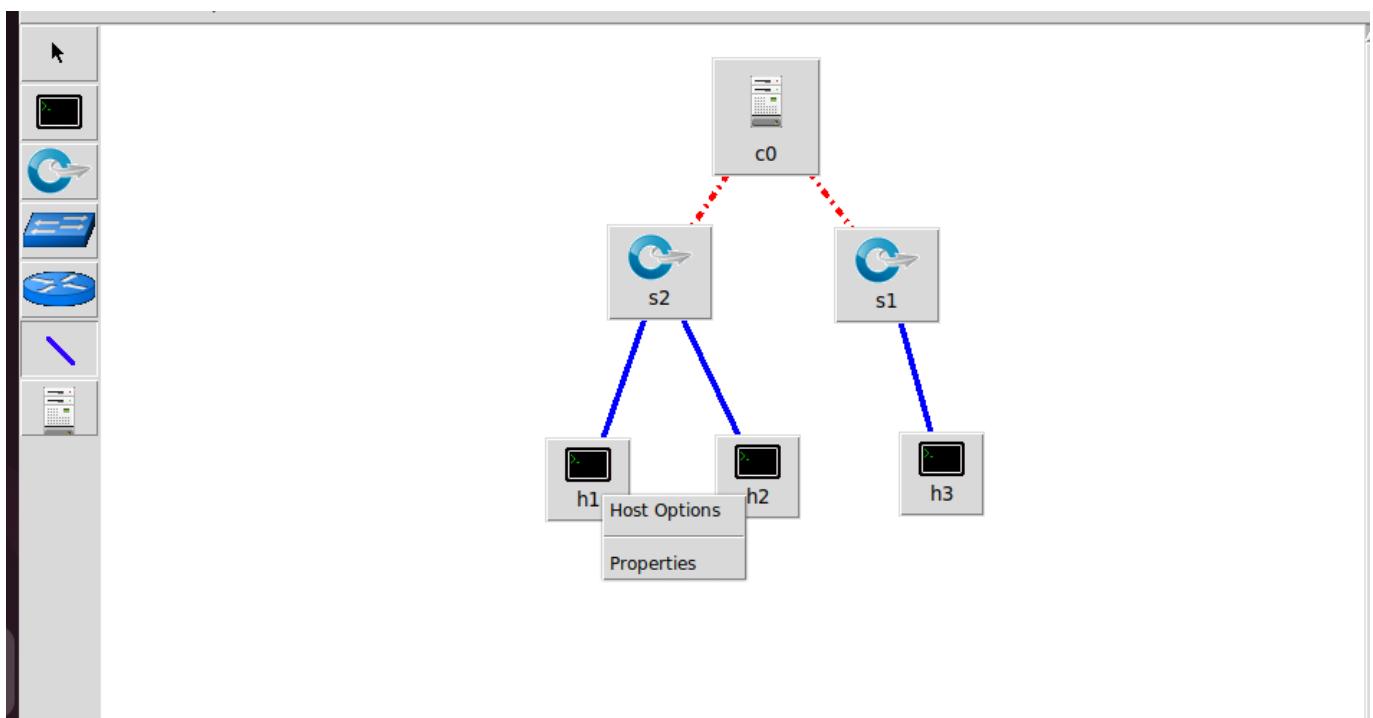
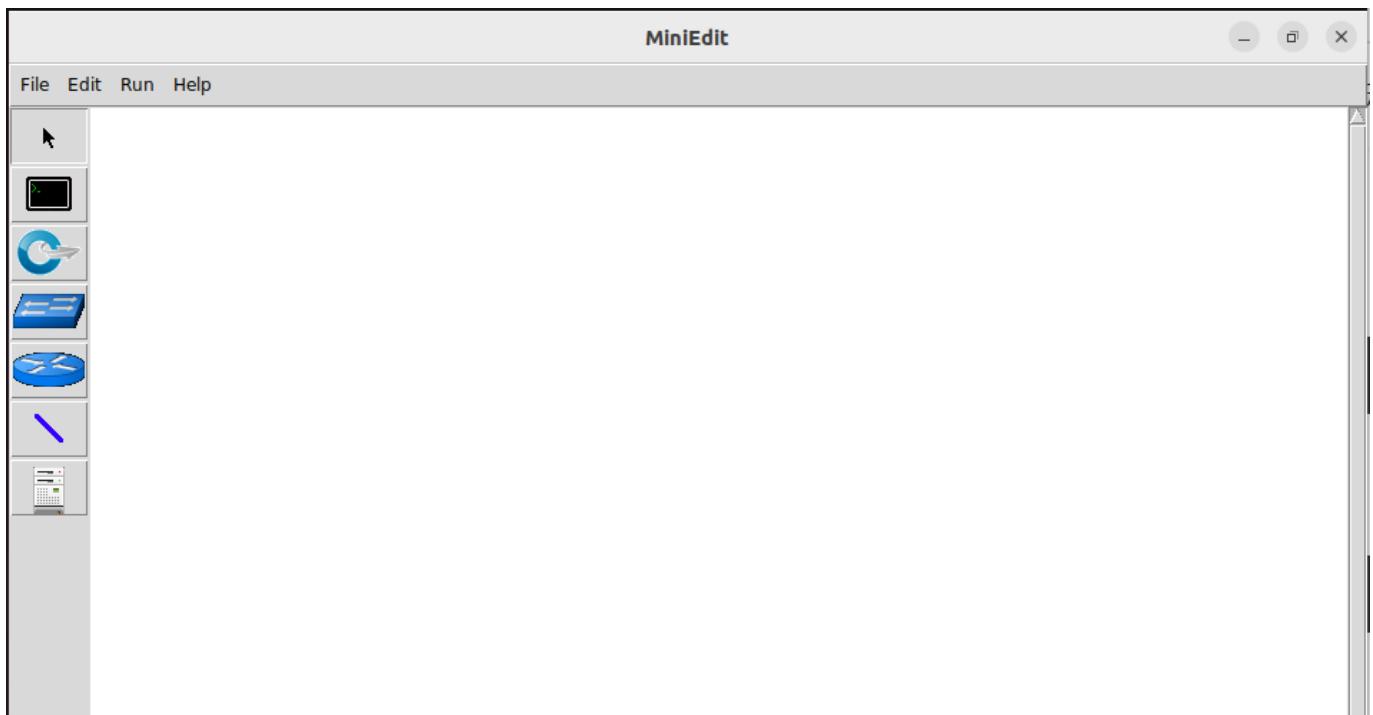
注意：执行miniedit.py，需要在桌面版系统下，或支持X11的情况下使用。Linux桌面版系统下可直接执行；远程使用Mininet虚拟机，需使用Xmanager或Xming。

### 运行miniedit.py

```
clusterSanity.py  controllers2.py  limit.py        multitest.py    scratchnetuser.py
consoles.py       intfoptions.py  linearbandwidth.py natnet.py      simpleperf.py
root@hyy-VirtualBox:/home/hyy/mininet/mininet/examples# ./miniedit.py
```

可能会出错，主要python版本和位置；

可以看到：



## 7. Mininet应用

# 应用1——Mininet中流表应用实战



## 案例目的：

掌握Open vSwitch下发流表操作；

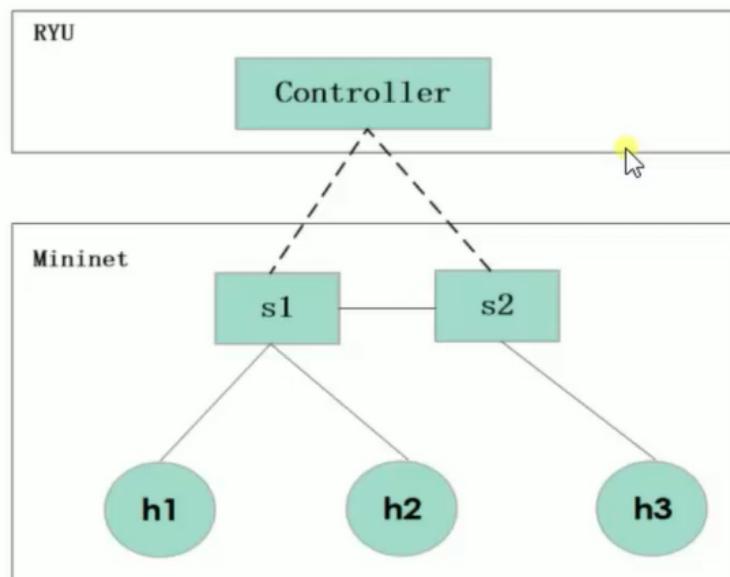
掌握添加、删除流表命令以及设备通信的原理。



Mininet创建一个默认树形拓扑并指定Mininet的控制器，进行基本的添加、删除流表操作，使网络实现网络通信和不通信。

*Operate  
flow tables*

## 网络拓扑



[Ubuntu中找不到yum及安装方法\\_sudo: yum: command not found-CSDN博客](#)

## 创建拓扑并连接RYU控制器



启动RYU：

```
# cd ~/ryu/ryu/app  
# ryu-manager simple_switch.py
```

Mininet远程连接ryu控制器：

## 查看流表



```
mininet> dpctl dump-flows ##查看静态流表
```

```
mininet> pingall  
*** ping: testing ping reachability  
h1 -> h2 h3  
h2 -> h1 h3  
h3 -> h1 h2  
*** Results: 0% dropped (6/6 received)  
mininet>
```

pingall后再次查看流表，有了流表后交换机根据流表进行数据包的转发使其通信，我们也可以人工的进行流表的新增、修改、删除等操作，在mininet网络系统下可直接输入命令。

## 添加流表

删除之前的所有流表：

```
mininet>dpctl del-flows
```

此时，流表为空，通过dpctl手动添加流表项，实现数据转发。

```
mininet>dpctl add-flow in_port=1,actions=output:2
```

```
mininet>dpctl add-flow in_port=2,actions=output:1
```

此时查看流表可以看到新的流表转发信息，同样可以使h1和h2之间ping通。

## 删除流表

如将删除条件字段中包含in\_port=1及in\_port=2的所有流表

```
mininet> sh ovs-ofctl del-flows s1 in_port=2
```

或

```
mininet> dpctl del-flows in_port=1
```

```
mininet> dpctl del-flows in_port=2
```

因为之前添加的1和2号端口的流表已被删除，使用dpctl dump-flows查看。

## 添加丢弃数据包的流表

例如让交换机丢弃从2号端口发来的所有数据包

```
mininet> dpctl add-flow in_port=2,actions=drop
```

```
mininet> dpctl dump-flows
*** s1 -----
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=18.760s, table=0, n_packets=0, n_bytes=0, idle_age=18, in_port=2 actions=drop
*** s2 -----
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=18.759s, table=0, n_packets=0, n_bytes=0, idle_age=18, in_port=2 actions=drop
```

增加这条流表以后，Mininet中主机之间将无法通信。

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
mininet>
```