Mininet 操作指南

By 北北（liubeibei789@gmail.com）

created: 02/13/2017

last edited: 05/01/2017

Contents

1. 虚拟机的设置：

Oracle VM VirtualBox Manager，下面是我 new 的两个虚拟机：
1.名字：mininet（我给它起的）是 2.2.1 版本，纯 Mininet，Berkerly 开发的
2.名字：MININET（我给它起的）是 2.1.0 版本，是 extended version，MiniNExT，USC 开发的

单击点中哪个虚拟机，再点菜单栏的黄色齿轮 setting，就是这个虚拟机对应的设置菜单。
大部分设置保持默认即可。需要注意的有：

Storage 标签下：

Controller: IDE：小写的 mininet 中，是 VBoxGuestAdditions.iso（这个是搞 shared folder 的时候添加的，如果不用 shared folder 则没有这个也行）；大写的 MININET 中，是 empty
Controller: SATA：是 mininet-vm-x86_64.vmdk（这个是从 Github 上下载的包解压后就有）
右侧的 Attributes 一栏：Name: IDE; Type: PIIX4，下面的 use host I/O cache 勾上

Network 标签下：

adapter 1:　enable network adapter 勾上，attached to: 选 bridged adapter
把 mininet 的 adapter 1 和宿主 ubuntu 操作系统的网络接口“桥接”起来，方便 mininet 连外网（指 www.google.com 等等）
Name: 选 enp5s0
下面的 advance 隐藏菜单：promiscuous mode: 默认是 deny，改选 allow all（这个是为了从宿主 ubuntu 的 terminal 中 ssh 进去）
adapter 2:　enable network adapter 勾上，attached to: 选 host-only adapter
是虚拟机和它 xterm 生成的 host 之间的通信接口
Name: 选 vboxnet
下面的 advance 隐藏菜单：promiscuous mode: 默认的 deny 即可（因为这个是虚拟机内部的，不涉及 ssh）
* host-only, NAT, bridged 三者的区别：
1）host-only：只给整个虚拟机 mininet 赋一个 IP 地址，这个 IP 地址只允许 mininet 内部的 device 用来和宿主 Ubuntu 通信，外部的其他主机不能和这个 ip 地址通信。
2）NAT（Network Address Transition）：分给虚拟机一个子网段，如宿主 ubuntu 是 192.168.6.1，则给虚拟机 mininet 分的一个子网段是 192.168.6.3（这个地址是刚才那个 192.168.6.1 的子集）。mininet 中的 device 可以访问外网，但外网不能访问 mininet 中的 device（存在保护）。虚拟机的所有活动，都看上去像是从宿主 ubuntu 发出的一样。
3）bridged：为虚拟机 mininet 构造一个存在于和宿主 ubuntu 一样地位的虚拟的节点，并被赋予一个独立的 ip 地址（如果允许 DHCP 的话）

注意：NAT 和 bridged 都是切切实实把 mininet 的 eth0（或 eth1,eth2 等），与宿主 ubuntu 的一个网卡连接起来，区别是 NAT 换内部地址，bridged 用独立 IP，这两者都只能建立一个这样的连接；而 host-only 是和宿主 ubuntu 之间建立的虚拟连接，可以建立多个，即如在启动前设置虚拟机 adapter 2 和 adapter 3 均为 host-only，启动后输入 sudo dhclient eth1，再输入 sudo dhclient eth2，就会有两个可以 ssh 登入的网卡。即 ssh -Y mininet@192.168.56.**102** 或 ssh -Y mininet@192.168.56.**101** 均可。当然实际操作的时候因为密钥的某些原因，只能设置其中一个为合法的，不能两个同时登。（此处没技术困难，按提示操作即可顺利切换）

Shared folders 标签下：

可以设置一个共享文件夹，宿主 ubuntu 和这个虚拟机之间的共同可以访问的文件夹
平时 mininet 中的操作，在根目录下比较方便，当需要和 ubuntu 传文件的时候，使用 shared folder

每次操作 shared folder 之前都要先同步：输入 sudo mount -t vboxsf share /mnt/folder
其中，“share”是在宿主 ubuntu 中的文件夹：Documents/share；
“folder”是在 mininet 中的文件夹：/mnt/folder
* 把需要传到宿主 ubuntu 的文件 MyFile.py，从 mininet 的根目录复制到 shared folder 里：
输入 sudo cp MyFile.py /mnt/folder
* 把 ubuntu 的 share 文件夹里的文件拿到 mininet 里用：
输入：sudo cp /mnt/folder/MyFile.py ./MyFile.py （~/表示根目录，./表示当前目录）
注意：/mnt/folder 中的文件可以随意访问，但读写权限是 read only，因此必须加 sudo

User interface 标签下：

可以设置显示的各种标签，发现哪些标签没有了可以在这里调

2. 启动虚拟机的操作：

双击要用的虚拟机，会弹出一个黑色界面的窗口，自动进行一大堆初始化之后，
登录：
Mininet-vm login: 输入 mininet 回车，password: 输入 mininet 回车（虽然两个虚拟机名字不同，一个大写一个小写，但那都是我自己给它起的而已，用户名和密码都是小写）

从宿主 ubuntu 中 ssh 进入 mininet（mininet 部分）：

Mininet-vm:~$ 先输入：ifconfig，显示当前存在的网络接口：

1.eth0: link encap: Ethernet
inet addr: 10.206.202.103

（这个 eth0 设置的是 bridged adapter，是虚拟机的网卡“桥接”宿主 ubuntu 的对外的网卡接口。这是个对外的地址，从其他的电脑（比如一个 windows 的笔记本）来试图连接这个虚拟机的时候，向这个地址发送连接请求。每次登录虚拟机，此地址有可能会变，需要与外界连接时注意先检查一下，在 windows 对应的 socket 连接程序的 config 文件中改好）

2.lo: link encap: Local Loopback
inet addr: 127.0.0.1（虚拟机的本地还回接口地址）

为了从宿主 ubuntu 的 terminal 中 ssh 进虚拟机（因为 terminal 中可以用鼠标复制粘贴更方便，而且有些需要图形界面的应用只有 ubuntu 才支持），我们再添加一个接口，输入：sudo dhclient eth1，没有什么现象发生。
再次输入 ifconfig，就会发现网络接口变为了三个：
（或者输入 ifconfig eth1，只看这一个网卡）

3.eth1: link encap: Ethernet
Inet addr: 192.168.56.101

（这个 eth1 是我们在设置中的 adapter 2，类型是 host-only adapter。就是虚拟机和宿主 ubuntu 之间的通信接口，比如从 terminal 中 ssh 进虚拟机时，就指定这个地址）

这些做完之后，就可以把这个黑色的窗口和 VM-Box 的窗口都最小化了。

从宿主 ubuntu 中 ssh 进入 mininet（ubuntu 部分）：

打开宿主 ubuntu 的 terminal：
beibei@beibei-OptiPlex-390:~$ 输入：ssh -Y mininet@192.168.56.101
（这个 mininet 是人家开发的虚拟机的名字，不随我自己起的 mininet 或 MININET 而改变。
192.168.56.101 是 host-only adapter 的地址，在添加 eth1 之后用 ifconfig 查看一下）
mininet@192.168.56.101’s password: 输入：mininet
显示：mininet@mininet-vm:~$ 就成功进入了虚拟机。之后的所有“在 mininet 中的命令”都是在这里输入的（还有一种叫做“CLI 中的命令”，是 sudo mn 之后搞出来的，注意区别二者）

ubuntu 和 mininet 的网络连接关系：

网络接口对应关系：

<table>
<tr><td colspan="2">ubuntu</td><td colspan="2">mininet</td></tr>
<tr><td colspan="2">beibei@beibei-OptiPlex-390:~$ ifconfig
（显示所有有 IP 地址（即正在运行，如 UP BROADCAST MULTICAST）的网卡，ifconfig -a 显示所有网卡，不管是不是正在运行，有没有 IP 地址）</td><td colspan="2">mininet@mininet-vm:~$ ifconfig</td></tr>
<tr><td>enp5s0
这些看上去 fancy 的名字其实就是厂商起的而已</td><td>10.206.201.150
这个是 ubuntu 对外的网卡，即使没有 mininet 这回事，别的主机想连接 ubuntu，也是向这个地址请求</td><td>eth0
(attached to bridged adapter: enp5s0)</td><td>NAT: 10.0.2.15

bridged: 10.206.201.201</td></tr>
<tr><td>vboxnet0</td><td>192.168.56.1
mininet 中的拓扑连接 remote controller 的时候，向这个地址请求（controller 是在这个地址上的）</td><td>eth1
(attached to host-only adapter: vboxnet0)</td><td>192.168.56.102
1.一进入 mininet，它默认是没有这个网卡的，要 sudo dhclient eth1 来通过 dhcp 协议动态获取一个 IP；
2.从 ubuntu 中 ssh 进入 mininet 的时候，就向这个地址请求连接</td></tr>
<tr><td>lo</td><td>127.0.0.1</td><td>lo</td><td>127.0.0.1</td></tr>
<tr><td>vmnet1</td><td>192.168.133.1</td><td>s1</td><td></td></tr>
</table>

host 连接外网 Internet:

Mininet 创建拓扑主要有两种方法：

1) 创建一个拓扑类，在命令行通过--topo 选项指定使用此拓扑：

如，建拓扑的时候，输入：sudo mn，或者 sudo mn --topo single,3 等等

在 CLI 中用 xterm h1 弹出来的小窗口里：

输入：ovs-vsctl show，显示出所有的接口：s1-eth1，s1-eth2，s1(internal)

2) Mininet 支持参数化拓扑，通过 python 代码创建一个灵活的拓扑结构，而且可以灵活实现额外的功能需求，为了 ping 外网，就是使用这种方式创建拓扑：

先用 miniedit 的 GUI（输入：./mininet/examples/miniedit.py）画一个拓扑，保存为 level-2 script，命名为：MyTopo.py，这样的拓扑 python script 中已经给 openvswitch 添加了和宿主 ubuntu 相连的 eth0，因此就不必再用 ovs-vsctl add-port s1 **eth0** 这样的命令添加网卡了

vim 打开 MyTopo.py 文件：（注意：拓扑文件的修改和运行都要求 root 权限，加 sudo）

```
info( '*** Starting network\n')
net.start()          #在启动了网络之后，添加下面 3 行
os.popen('ovs-vsctl add-port s1 eth0')          #绑定 s1 和宿主 ubuntu 的 eth0 接口
h1.cmd('dhclient '+h1.defaultIntf().name)       #给 h1 的网口用 DHCP 动态分配 IP
h2.cmd('dhclient '+h2.defaultIntf().name)       ##给 h2 的网口用 DHCP 动态分配 IP
CLI(net)
net.stop()
```

在 CLI 中用 xterm h1 弹出来的小窗口里：
输入：ovs-vsctl show，显示出所有的接口：s1-eth1，s1-eth2，s1，**eth0**

在 xterm 弹出的 h1 小窗中：
1）ping 外网：ping www.google.com，可以 ping 通
2）ping 10.206.201.150（宿主 ubuntu 的网卡 enp5s0，相当于 ubuntu 的 eth0）可以 ping 通，因为我们就是把 mininet 的 eth0 和宿主 ubuntu 的网卡 enp5s0 桥接了
3）ping 192.168.56.1（宿主 ubuntu 的网卡 vboxnet0，相当于 ubuntu 的 eth1）不能 ping 通，因为并没有和这个网卡相连
4) ping 10.206.201.201（虚拟机 mininet 的网卡 eth0），不能 ping 通？？？

另外，上述用 python script 的方法建拓扑，在我们人为添加的 3 行中，从 DHCP server 为 h1 和 h2 申请了动态 ip（如 h1: 10.206.202.145，h2: 10.206.202.132），两者可以互 ping。而且为了 ping 通外网，h1 和 h2 必须拥有独立的 ip，10.0.0.1 和 10.0.0.2 是不行的。

3. Mininet 的操作：

先建立拓扑：

最简单的：输入：sudo mn，可以建立 1 个 switch，2 个 hosts 的拓扑结构

几个例子（网上找的）：

1.[单交换机] 创建具有 1 个交换机，交换机上连接 3 台主机的网络拓扑结构

输入：sudo mn --arp --topo single,3 --mac --switch ovsk --controller=remote,ip=192.168.56.**1**,port=6633

（注意:controller 和 ip 之间没空格!）

（-mac：自动设置 MAC 地址，MAC 地址与 IP 地址的最后一个字节相同；-arp：为每个主机设置静态 ARP 表，例如：主机 1 中有主机 2 和主机 3 的 IP 地址和 MAC 地址 ARP 表项，主机 2 和主机 3 依次类推；-switch：使用 OVS 的核心模式；-controller：使用远程控制器，可以指定远程控制器的 IP 地址和端口号，如果不指定则默认为 127.0.0.1 和 6633（相当于 mininet 给你配个默认的 controller），但注意！如果写了--controller remote 没有指定控制器的话，是 ping 不通的！）

2. [两个线性连接的交换机] 创建具有 2 个交换机，两个交换机下面个连一个主机，交换机之间再互连起来

输入：sudo mn --topo linear --switch ovsk --controller remote

3. [树形拓扑]

输入：mn --topo tree,depth=2,fanout=3

192.168.56.**1** 这个地址是宿主 ubuntu 的 host-only adapter 的对内部（给虚拟机看）的地址。为什么不是 eth1 的地址 192.168.56.**101** 呢？因为 101 这个是从宿主 host 登虚拟机的时候（ssh）用的，而 192.168.56.**1** 这个是从虚拟机来找宿主 ubuntu 的。我们现在是虚拟机 mininet 要寻找 controller，controller 是运行在宿主 ubuntu 上的，因此使用这个地址。

怎样查看这个地址呢？在宿主 ubuntu 中查看网络接口：

重新打开一个 Ubuntu 的 terminal（注意不是当前这个已经 ssh 进 mininet 了的窗口），输入 ifconfig 回车，显示 3 个网卡：enp5s0, lo, vboxnet0

enp5s0 是虚拟机中的 eth0（bridged adapter）对应的宿主接口，inet addr: 10.206.201.150

vboxnet0 是是虚拟机中的 eth1（host-only adapter）对应的宿主接口，inet addr: **192.168.56.1**，这个就是我们需要的连接 controller 的地址啦。（controller 怎么启动后面讲）

输入了建立拓扑的命令 sudo mn blabla…之后，显示。。。

前面的标识变成 mininet>，就进入 mininet 的 CLI（command line interface），在这里面输入的命令就是直接可以操作 mininet 创建的拓扑的了。但我们一般不在这个命令行中敲，而是写好的 python script 的.py 文件在 mininet 中直接运行：

cd 文件所在的文件夹

python 文件名.py

（或 python 带路径的文件名.py）

注意！在实验完之后，记得清除拓扑：sudo 空格 mn 空格-c，否则下次再启动 mininet 的时候，原来的拓扑还在。

启动控制器：

（在 ubuntu 的 terminal 中）先 cd（change directory）到控制器所在的文件夹（从 github 上下载解压即可），然后运行特定的 python 程序文件。

pox:

cd pox，然后输入./pox.py 空格 pox.forwarding.hub（相当于.py 文件的运行参数）

会显示：INFO:core:pox 0.2.0 (carp) is up，说明控制器已经启动，最小化即可，去 mininet 中建立拓扑

ryu:

运行传统二层交换机：cd ryu/ryu/app，然后输入 ryu-manager simple_switch.py　　**？？？ryu 不是控制器吗？**

cd ryu/ryu/app/gui_topology，然后输入 ryu-manager gui_topology.py

4. 我们项目的整体仿真系统连接：

windows 1 → mininet 1 → host 1 → host 2 → mininet 2 → windows 2，然后再逆向传回，形成一个循环。

	windows 1	mininet 1	host 1	host 2	mininet 2	windows 2
接收 IP 地址	127.0.0.1	10.206.201.201	10.0.0.1	10.0.0.2	10.206.201.201	127.0.0.1
监听端口号	2222	4444	6666	6666	5555	3333
发送 IP 地址	10.206.201.201	10.144.154.181	10.0.0.2	10.0.0.1	10.144.154.181	10.206.201.201
发送端口号	4444	2222	6666	6666	3333	5555
程序名	comm_win_power.py	mininet1.py	host1.py	host2.py	mininet2.py	comm_win_water.py
recv 文件名	powerInFile	h1toWater	h1toPower	h2toWater	h2toPower	waterInFile
trans 文件名	powerOutFile	h1toPower	h1toWater	h2toPower	h2toWater	waterOutFile

注意实验开始前把笔记本的防火墙禁用：
this PC → open control plane → system security → windows firewall → turn windows firewall on and off，把 private 和 public network 的两个防火墙都关掉（实验完再打开）

mininet 1，host 1，host 2，mininet 2 是在 linux 系统（ubuntu）上运行的，windows 1 和 windows 2 是在 windows 上运行的。

mininet 1 & mininet 2:
我们希望 mininet 1 和 mininet 2 分别在各自的 terminal 上运行，于是在 ubuntu 中开两个新的 terminal，ssh 进入 mininet，分别运行 mininet1.py 和 mininet2.py 文件（代码参见 github: https://github.com/liubeibei789/Mininet-Windows-Comm.）

host 1 & host 2:
mininet 内部的两个 host，我们用 xterm 命令弹出 h1 和 h2 各自的窗口
Xterm: 新弹出窗口的命令。比如在 CLI（就是 sudo mn 搞出来的那个以 mininet>开头的界面）中输入：xterm h1，就会弹出 h1 的窗口，h2，s1 同理

先 ping 测试 host 1 和 host 2 之间底层连接是否通畅：
在 h1 小窗中输入：ping 10.0.0.2（host 2 的地址）
若指定 ping 的包的数量，如 3 个：ping -c 3 10.0.0.2

在可以 ping 通的前提下，就可以互相传递文件：
接收方：在 xterm h1 出来的小窗口中输入：nc -l-p 1234 >ReceivedFile.txt
（l 是字母 L 的小写，1234 是随意一个 4 位端口号，ReceivedFile 是接收的文件名，类型 zip 等也可）
发送方：在 xterm h2 出来的小窗口中输入：nc-w 3 10.0.0.1 1234 <SentFile.txt
（10.0.0.1 是接收方 host 1 的地址，1234 是接收方端口号，SentFile 是发送的文件名，类型不限）

以上是命令方式在 mininet 中的 host 之间传文件。但是，我们的项目不通过这种方式传递文件，而是运行写好的 python script: “host1.py”和”host2.py”（代码参见 github: https://github.com/liubeibei789/Mininet-Windows-Comm.）

windows 1 & windows 2:
cd 到放 python 程序文件的文件夹，输入 python comm_win_power.py（或其他程序名字.py）

* 注意：这些 python 程序的启动顺序没有太大关系（因为有重试机制保障）：
每个程序都是由 receive()和 transmit()函数构成的，分别占用一个线程。整个系统是个循环，最开始的 comm_win_power.py 程序产生那个要在系统中循环的文件，并发送给下一个模块接收。任何一个 transmit()模块在向它想传递到的 server 提出连接请求时，若出现 server 没有准备好的情况，它会 wait()一秒再重新发送请求。transmit()模块本身的触发是来自“收到某些文件”的信号，因此可以自动触发下一个模块。

5. 系统性能评估及可视化：

wireshark:

用途是：在 mininet 中 capture traffic

在 mininet 中输入：sudo wireshark &>/dev/null &，就可以启动 wireshark 软件了，可以监控不同的端口，mininet 中的流量；

在 mininet 中输入：sudo wireshark & 则是监控宿主 ubuntu 的流量，！！而且注意不是在 ubuntu 的 terminal 中输入，而是在 mininet 中输入启动命令

注意：我的 wireshark 软件是在 ubuntu 自带的 software center 中安装的（不是安装在 mininet 中通过命令安装的软件）；另外我当时设置了 user 权限，所以现在每次启动必须 sudo，否则无法显示可抓包端口

miniedit:

图形界面的显示 mininet 拓扑的软件。因为需要支持图形界面，miniedit 只能在宿主 Ubuntu 上运行，即必须是从 ubuntu 的 terminal 中 ssh 进虚拟机，才能启动 miniedit

miniedit 可以通过 GUI 来手动制作一个拓扑，还可以转化为代码。但我们做的项目后面功能都会比较复杂，customized 的地方比较多，则进一步修改这个 python 程序即可。

命令：./mininet/examples/miniedit.py

连接完拓扑之后，点 file→export level 2 script，即可保存成产生拓扑的 python 代码（注意保存时点一下左侧的 home 标签，有时候存到 computer 里不知道是到哪里去了）

6. 一些常用的 linux 命令行操作：

文件：

回到上一级目录：cd ../
回根目录：cd ~/ 或 cd ~
删文件：rm -f 文件名.py
删文件夹：rm -rf 文件夹名.py （注意 linux 没有回收站）
新建文件：vim 文件名
创建目录：mkdir 目录名
复制粘贴：cp file1 file2（两个文件都是带目录的）

vim:

vim 文件名.py 回车，即进入编辑页面。但此时还不能编辑。
先按 i 进入插入模式。注意！在粘贴时，一定记得先进入插入模式再粘贴，否则会乱。
连续按两次 dd，删除整行。
编辑完毕，先按 esc 来推出编辑模式，然后根据你是否修改了文件内容，有下面几种选项：
:q 退出
:q! 强制退出
:wq 保存并退出

其他：

help：默认列出所有命令文档，后面加命令名将介绍该命令用法
dump：打印节点信息
gterm：给定节点上开启 gnome-terminal。注：可能导致 mn 崩溃
xterm：给定节点上开启 xterm
intfs：列出所有的网络接口
iperf：两个节点之间进行简单的 iperf□TCP 测试
iperfudp：两个节点之间用制定带宽 udp 进行测试
net：显示网络链接情况
noecho：运行交互式窗口，关闭回应（echoing）
pingpair：在前两个主机之间互 ping 测试
source：从外部文件中读入命令
dpctl：在所有交换机上用 dptcl 执行相关命令，本地为 tcp□127.0.0.1:6634
link：禁用或启用两个节点之间的链路
nodes：列出所有的节点信息
pingall：所有 host 节点之间互 ping
py：执行 python 表达式
sh：运行外部 shell 命令
quit/exit：退出

批量移动文件：

```
#------------move_files.py---------------
import os
```

```
import sys
DefaultFolder = '~/comm'    #如果多个文件要移到同一个文件夹里去，则设置一个默认目标文件夹
FileName = sys.argv[1]
DestFolder = sys.argv[2] if len( sys.argv ) > 2 else DefaultFolder
        #sys.argv 下标是从 0 开始的，sys.argv[0]是带路径文件名，[1],[2]是输入参数
print 'FileName:'+FileName              #打印出两个输入参数
print 'DestFolder:'+DestFolder

os.popen('cp '+FileName+' '+DestFolder+'/'+FileName)      #复制到目标文件夹
os.popen('rm '+FileName)                                  #删除当前文件
#---------------------------------------------
```

运行的时候，输入：python move_files.py host.py (argv[1]) ~/comm (argv[2])