Mininet 代码分析

最新版: yeasy@github

更新历史:
V0.3: 2013-11-18
    完成运行文件分析。
V0.2: 2013-11-15
    完成库文件分析。
V0.1: 2013-10-11
    完成代码结构分析。

1. 源代码结构

1.1. 运行相关

bin/mn

主运行文件，安装后执行 mn 即调用的本程序，是 python 程序。

mnexec.c

执行一些快速命令，比如关闭文件描述符等，是 C 程序，编译后生成二进制文件 mnexec 被 python 库调用。

1.2. Install 相关

INSTALL：安装说明

setup.py：安装 python 包时候的配置文件，被 Makefile 中调用。

debian/：生成 deb 安装包时的配置文件。

1.3. 核心代码

核心代码基本都在 mininet/子目录下。

注：最新的 2.1.0 版本，核心 python 代码仅为 4675 行。

```
find mininet  -name "*.py" | xargs cat | wc -l
```

1.4. 说明文件等

CONTRIBUTORS：作者信息

README.md：主说明文件

doc/doxygen.cfg：执行 doxygen 生成文档时的配置文件。

1.5. 其他文件

LICENSE

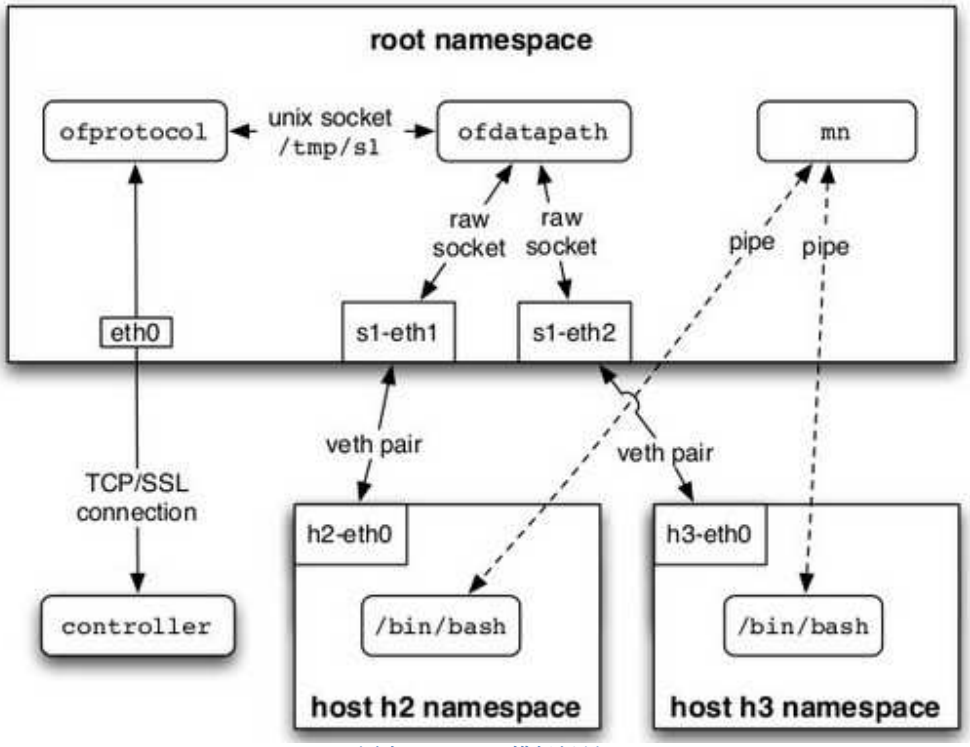
custom/ 目录下可以放一些用户自定义的 python 文件，比如自定义的拓扑类等。

test/目录下是一些测试的例子。

util/目录下是一些辅助文件，包括安装脚本、文档辅助生成等。

1.6. 整体功能逻辑

整体上来看，mininet 作为一个基于 python 的网络模拟工具，可以分为两大部分：python 库和运行文件。前者提供对网络中元素进行抽象和实现，例如定义主机类来表示网络中的一台主机。后者则基于这些库完成模拟过程。一个典型的场景如下图所示。



图表 1 Mininet 模拟场景

2. 库文件分析

2.1. mininet.cli 模块

包括 CLI 类。

提供对 CLI 的支持，创建 mininet 的 bash，接受通过 bash 传输的 mininet 命令，形成可以进行交互的 mininet 命令行环境。

2.2. mininet.link 模块

描述链路相关的接口和连接。包括 Intf 类、Link 类、TCIntf 类和 TCLink 类。

2.2.1. mininet.link.Intf

表示基本的网络接口。

2.2.2. mininet.link.Link

表示基本的一条链路，最基本的链路在 mininet 中其实就是一对 veth 接口对。

2.2.3. mininet.link.TCIntf

被 TC（linux 下的 traffic control 的工具）自定义的接口，可以配置包括带宽、延迟、丢包率、最大队列长度等参数。

2.2.4. mininet.link.TCLink

表示一对对称的 TC 接口连接到一起。

2.3. mininet.net 模块

主要包括 Mininet 和 MininetWithControlNet 两个类。

2.3.1. mininet.net.Mininet

模拟一个 mininet 中的网络，包括拓扑、交换机、主机、控制器、链路、接口等。

其中最主要的部分是 build()函数，依次执行：根据拓扑创建网络，配置网络名字空间，配置主机的 ip、mac 等信息，检查是否启动 xterm，是否配置自动静态 arp 等。

2.3.2. mininet.net.MininetWithControlNet

继承自 Mininet 类，主要用于在使用用户态 datapath 的时候模拟一个控制器网络，即连接用户态的交换机和用户态的控制器。

2.4. mininet.log 模块

利用 logging 包，主要提供进行 log 相关的功能，包括三个类：MininetLogger、Singleton、StreamHandlerNoNewline。

2.4.1. mininet.log.MininetLogger

自定义的 logger 类。

提供输出 log、配置 LogLevel 功能。

2.4.2. mininet.log.Singleton

软件设计模式，限定所创建的类只能有一个实例。供 MininetLogger 使用。

2.4.3. mininet.log.StreamHandlerNoNewline

自动添加换行和对流进行格式化，供 MininetLogger 使用。

2.5. mininet.node 模块

这个模块表示网络中的基本元素，十分关键。

2.5.1. mininet.node.Node

表示一个基本的虚拟网络节点。在实现上其实就是在网络名字空间中的一个 shell 进程，可以通过各种管道进行通信。该类是模块中其他类的根本，其它类都是直接或间接继承。

节点包括名称、是否在网络名字空间、接口、端口等可能的属性。

2.5.2. mininet.node.Host

表示一个主机节点，此处跟 Node 类相同。

2.5.3. mininet.node.Controller

继控制器基类。表示一个控制器节点。包括 ip 地址、端口等。主要方法包括启动和停止一个控制器。

2.5.4. mininet.node.NOX

表示一个 NOX 控制器（需要系统中实现安装了 NOX）。

2.5.5. mininet.node.OVSController

表示一个 ovs-controller（需要系统中实现安装了 ovs-controller）。

2.5.6. mininet.node.RemoteController

表示一个在 mininet 控制外的控制器，即用户自己额外运行了控制器，此处需要维护连接的相关信息。

2.5.7. mininet.node.CPULimitedHost

继承自 Host 类，通过 cgroup 工具来对 cpu 进行限制。

2.5.8. mininet.node.Switch

表示一个交换机的基类。运行在 root 名字空间。主要包括 dpid、listenport 等属性。

2.5.9. mininet.node.IVSSwitch

表示一台 indigo 交换机（需要系统中已存在）。

2.5.10. mininet.node.OVSLegacyKernelSwitch

传统的 openvswitch 交换机，基于 ovs-openflowd。不推荐。

2.5.11. mininet.node.OVSSwitch

表示一台 openvswitch 交换机（需要系统中已经安装并配置好 openvswitch），基于 ovs-vsctl 进行操作。

2.5.12. mininet.node.UserSwitch

用户态的 openflow 参考交换机，即 ofdatapath。不推荐。

2.6. mininet.topo 模块

维护网络拓扑相关的信息。

2.6.1. mininet.topo.MultiGraph

表示一个图结构。类似于 networkx 中的图 G(V,E)的概念。主要维护节点、边信息。

2.6.2. mininet.topo.Topo

拓扑基类，默认的拓扑图被 multigraph 类维护，此外还包括节点、连接等信息。

2.6.3. mininet.topo.LinearTopo

表示一个线行拓扑，交换机连接成一条链，每个交换机上挂载相等个数的主机。

2.6.4. mininet.topo.SingleSwitchTopo

单个交换机上挂载若干主机，主机序号按照从小到大的顺序依次挂载到交换机的各个端口上。

2.6.5. mininet.topo.SingleSwitchReversedTopo

单个交换机上挂载若干主机，主机序号按照从大到小的顺序依次挂载到交换机的各个端口上。

2.7. mininet.topolib

提供用户自己创建复杂拓扑相关的库，目前仅包括一个 Tree 拓扑。

2.7.1. mininet.topolib.TreeTopo

树拓扑类，给定深度和广度可以自己生成相应的标准树拓扑。

2.8. mininet.moduledeps 模块

定义几个对 linux 系统中内核模块进行操作的函数，包括列出模块 lsmod，移除模块 rmmod，探测模块 modprobe 和处理模块的依赖等。

2.9. mininet.term 模块

支持 term 相关的命令，例如在主机上创建一个 xterm。实现依赖于 socat 和 xterm。

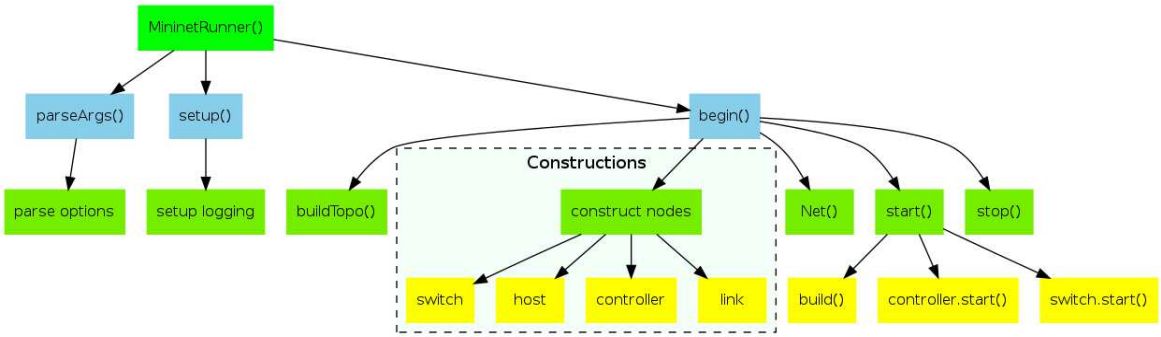
3. 运行文件分析

3.1. mn 脚本

该脚本定义了一个 MininetRunner 类，用来表示模拟网络的主程序。

主要过程是创建一个 MininetRunner()实例，依次解析传入参数，进行初始化后开启网络。

整体过程如下图所示。



图表 2 mn 脚本主要过程