Mininet 与 Ryu 控制器部署及流表操作指南

何明毅王子涵

June 10, 2025

Abstract

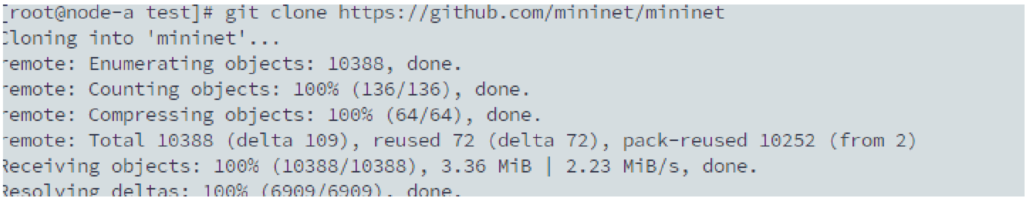
本文档详细描述了 Mininet 网络仿真平台的安装过程，Ryu 控制器的配置方法，以及通过 RESTful API 操作 OpenFlow 流表的完整流程。包含 Mininet 安装、Ryu 依赖配置、控制器启动、网络连通性测试和流表增删查改等关键操作步骤。

1 Mininet 安装与配置

1.1 下载 Mininet

通过 Git 克隆 Mininet 官方仓库：

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



```
[root@node-a test]# git clone https://github.com/mininet/mininet
Cloning into 'mininet'...
remote: Enumerating objects: 10388, done.
remote: Counting objects: 100% (136/136), done.
remote: Compressing objects: 100% (64/64), done.
remote: Total 10388 (delta 109), reused 72 (delta 72), pack-reused 10252 (from 2)
Receiving objects: 100% (10388/10388), 3.36 MiB | 2.23 MiB/s, done.
Resolving deltas: 100% (6909/6909), done.
```

Figure 1: Mininet 克隆过程

1.2 安装 Mininet

运行安装脚本进行完整安装：

```
cd mininet/util/
./install.sh -a
```

验证安装版本：

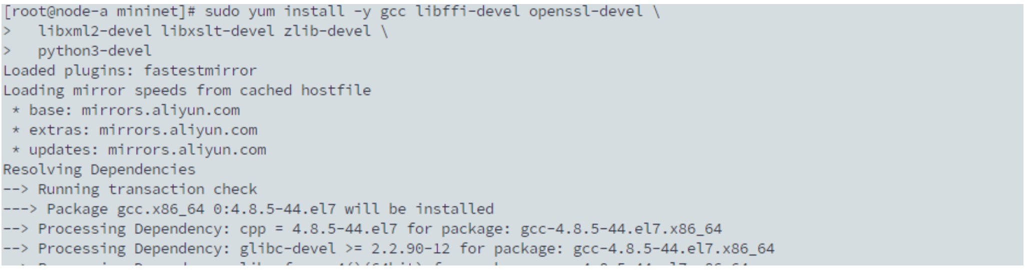
```
mn --version
# 输出: 2.3.1b4
```

2 Ryu 控制器安装

2.1 安装依赖包

安装 Ryu 所需的系统依赖：

```
sudo yum install gcc libffi-devel openssl-devel \
libxml2-devel libxslt-devel python3-devel
```



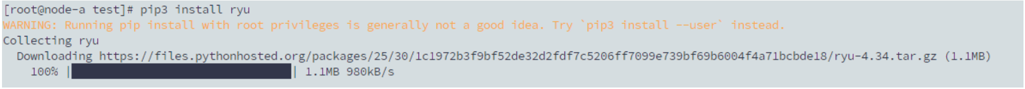
```
[root@node-a mininet]# sudo yum install -y gcc libffi-devel openssl-devel \
>   libxml2-devel libxslt-devel zlib-devel \
>   python3-devel
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
Resolving Dependencies
--> Running transaction check
---> Package gcc.x86_64 0:4.8.5-44.el7 will be installed
--> Processing Dependency: cpp = 4.8.5-44.el7 for package: gcc-4.8.5-44.el7.x86_64
--> Processing Dependency: glibc-devel >= 2.2.90-12 for package: gcc-4.8.5-44.el7.x86_64
```

Figure 2: Ryu 安装

2.2 使用 pip 安装 Ryu

通过 Python 包管理器安装 Ryu：

```
pip install ryu
```



```
[root@node-a test]# pip3 install ryu
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting ryu
  Downloading https://files.pythonhosted.org/packages/25/30/1c1972b3f9bf52de32d2fdf7c5206ff7099e739bf69b6004f4a71bcbde18/ryu-4.34.tar.gz (1.1MB)
    100% |████████████████████████████████| 1.1MB 980kB/s
```

Figure 3: 使用 pip 安装

3 RESTful 控制操作

3.1 启动 Ryu 控制器

在 Terminal 1 中启动 Ryu 控制器并加载 REST 应用：

```
ryu-manager ryu.app.ofctl_rest
# 输出: (10582) wsgi starting up on http://0.0.0.0:8080
```

```
(108582) wsgi starting up on http://0.0.0.0:8080
```

Figure 4: 启动 Ryu 控制器并加载 OFCTL REST 应用

3.2 启动 Mininet 网络

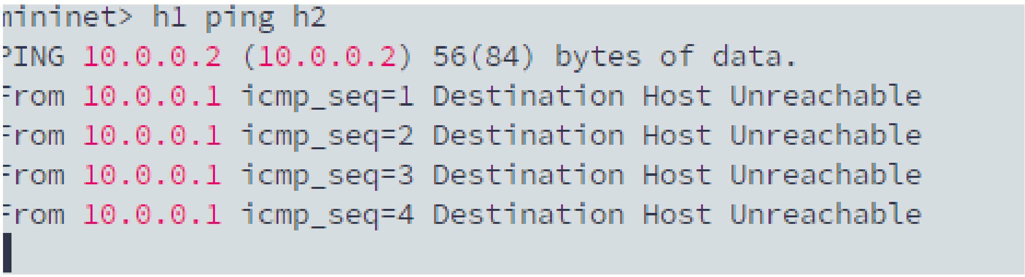
在 Terminal 2 中启动 Mininet 并连接到 Ryu 控制器：

```
mn --topo single,2 --controller=remote,ip=127.0.0.1,port=6653
```

3.3 验证初始状态

测试网络初始连通性：

```
mininet> h1 ping h2
# 输出: Destination Host Unreachable (初始无流表状态)
```



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
```

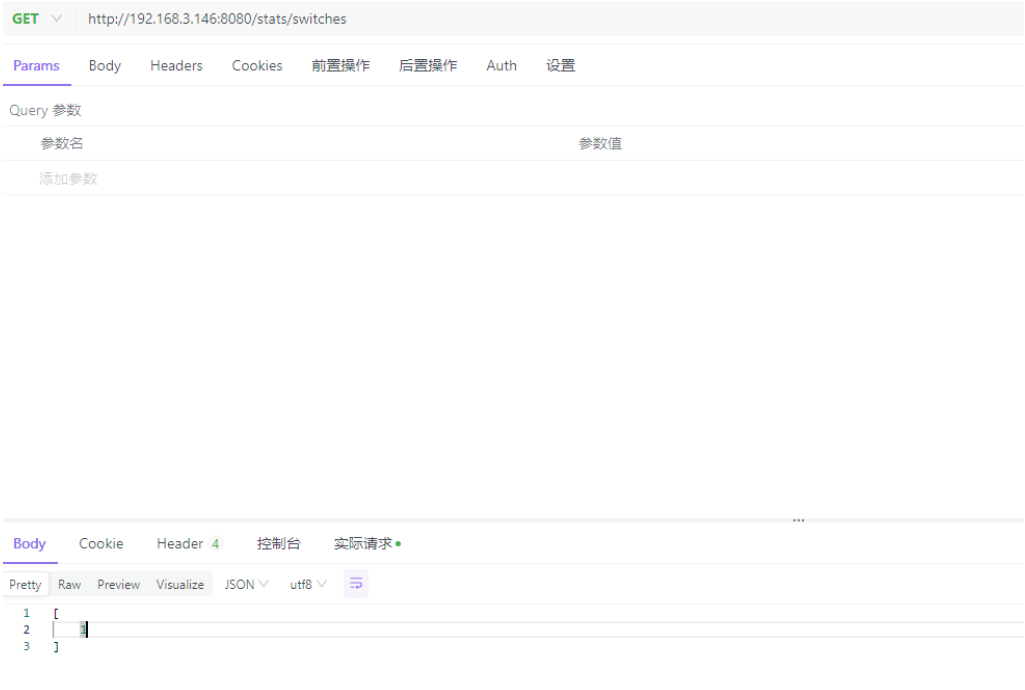
Figure 5: 测试网络初始连通性

4 流表操作 API

4.1 获取交换机信息

GET 请求获取交换机信息:

```
GET http://127.0.0.1:8080/stats/switches
```



GET http://192.168.3.146:8080/stats/switches

```
[
    1
]
```

Figure 6: GET 请求获取交换机信息

4.2 查看流表

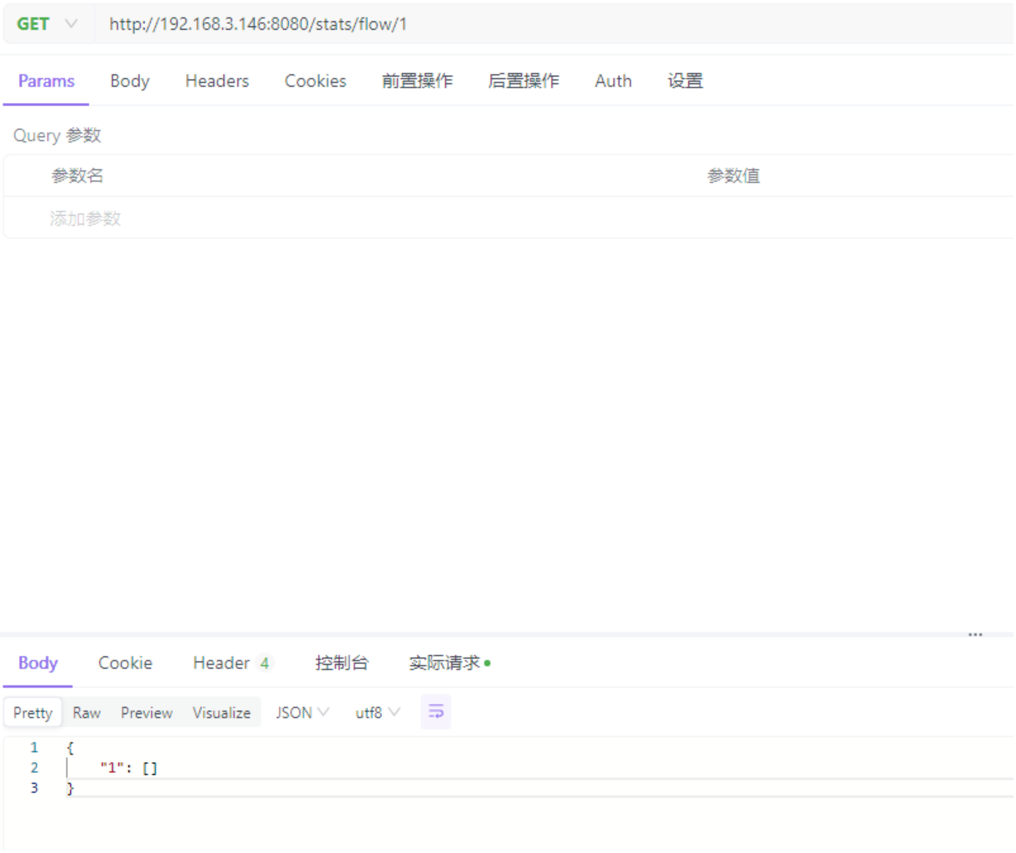
GET 请求查看当前流表:

```
GET http://127.0.0.1:8080/stats/flow/<dpid>
# <dpid>替换为实际交换机ID
```

4.3 添加流表项

添加两条流表项使 h1 和 h2 可以通信:

POST 请求 1:



GET http://192.168.3.146:8080/stats/flow/1

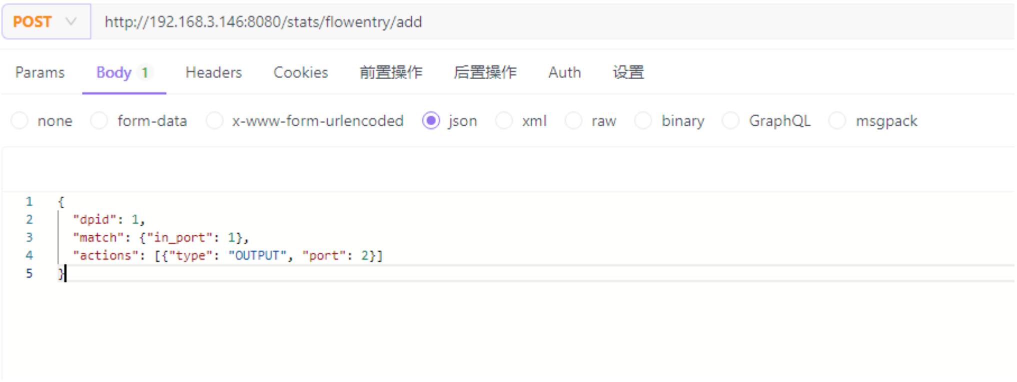
```
{
    "1": []
}
```

Figure 7: GET 请求查看当前流表

```
URL: http://127.0.0.1:8080/stats/flowentry/add
Method: POST
Body:
{
  "dpid": 1,
  "priority": 1,
  "match": {"in_port": 1},
  "actions": [{"type": "OUTPUT", "port": 2}]
}
```

POST 请求 2:

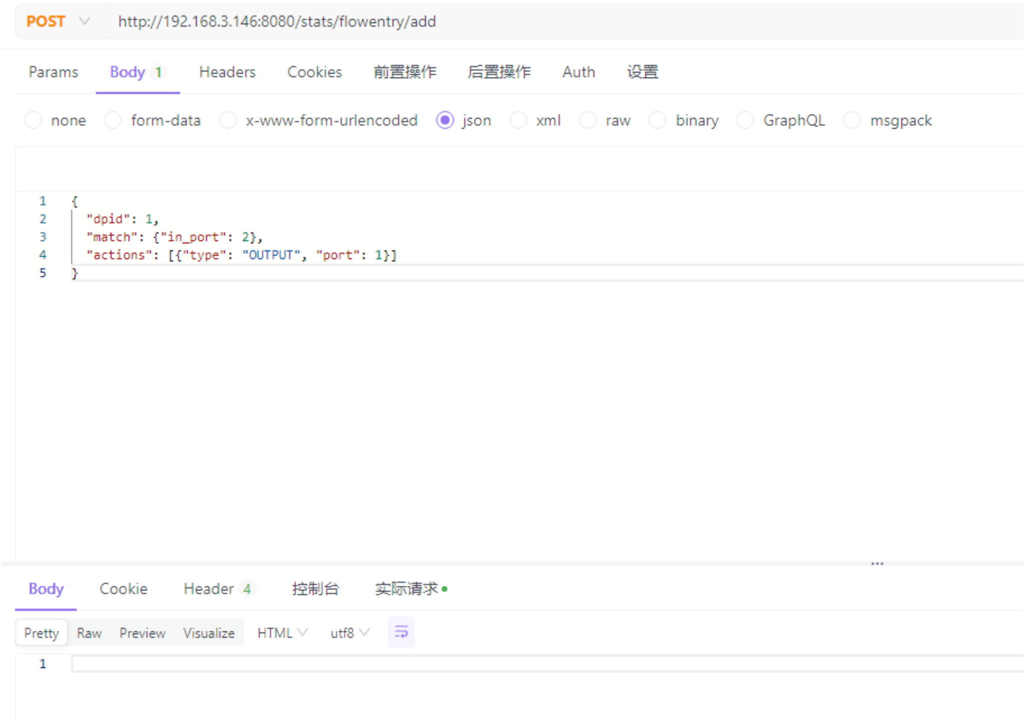
```
URL: http://127.0.0.1:8080/stats/flowentry/add
Method: POST
Body:
{
  "dpid": 1,
  "priority": 1,
  "match": {"in_port": 2},
  "actions": [{"type": "OUTPUT", "port": 1}]
}
```



POST http://192.168.3.146:8080/stats/flowentry/add

```
{
  "dpid": 1,
  "match": {"in_port": 1},
  "actions": [{"type": "OUTPUT", "port": 2}]
}
```

Figure 8: 添加流表项 1



POST http://192.168.3.146:8080/stats/flowentry/add

```
{
  "dpid": 1,
  "match": {"in_port": 2},
  "actions": [{"type": "OUTPUT", "port": 1}]
}
```

Figure 9: 添加流表项 2

4.4 验证流表生效

再次测试连通性并查看流表:

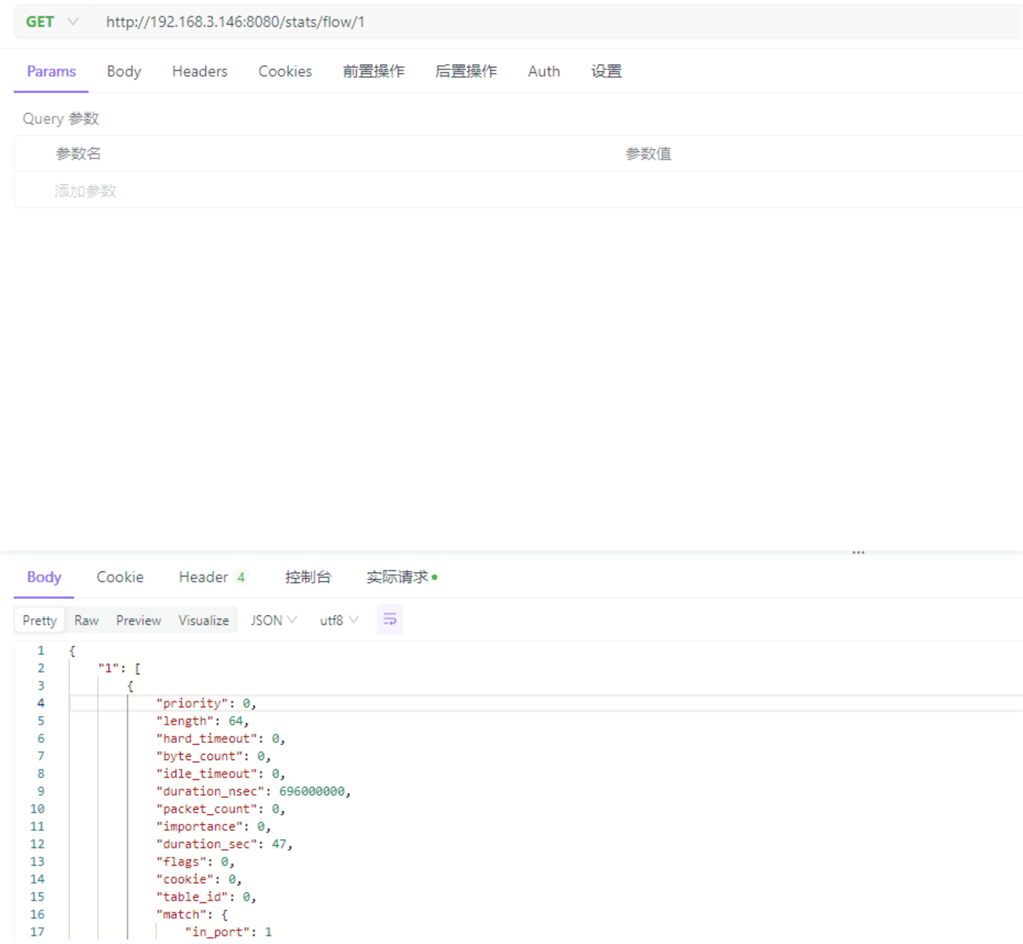
```
mininet> h1 ping h2
# 应显示正常ping响应

GET http://127.0.0.1:8080/stats/flow/1
# 显示添加的流表项
```

4.5 删除流表项

使用 DELETE 方法清除所有流表:

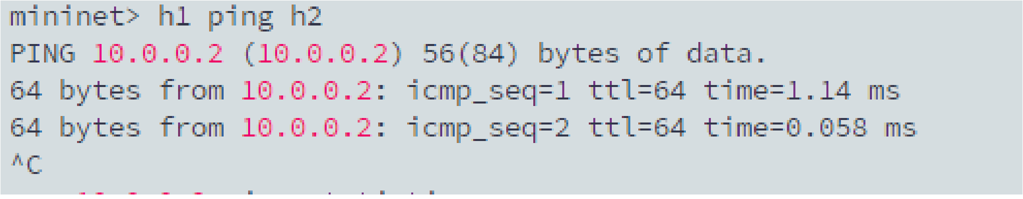
```
DELETE http://127.0.0.1:8080/stats/flowentry/clear/1
```



http://192.168.3.146:8080/stats/flow/1

```
{
    "1": [
        {
            "priority": 0,
            "length": 64,
            "hard_timeout": 0,
            "byte_count": 0,
            "idle_timeout": 0,
            "duration_nsec": 696000000,
            "packet_count": 0,
            "importance": 0,
            "duration_sec": 47,
            "flags": 0,
            "cookie": 0,
            "table_id": 0,
            "match": {
                "in_port": 1
```

Figure 10: 查看流表



```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.14 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.058 ms
^C
```

Figure 11: 流表生效后的 ping 测试

5 问题排查

5.1 删除流表后无法连接

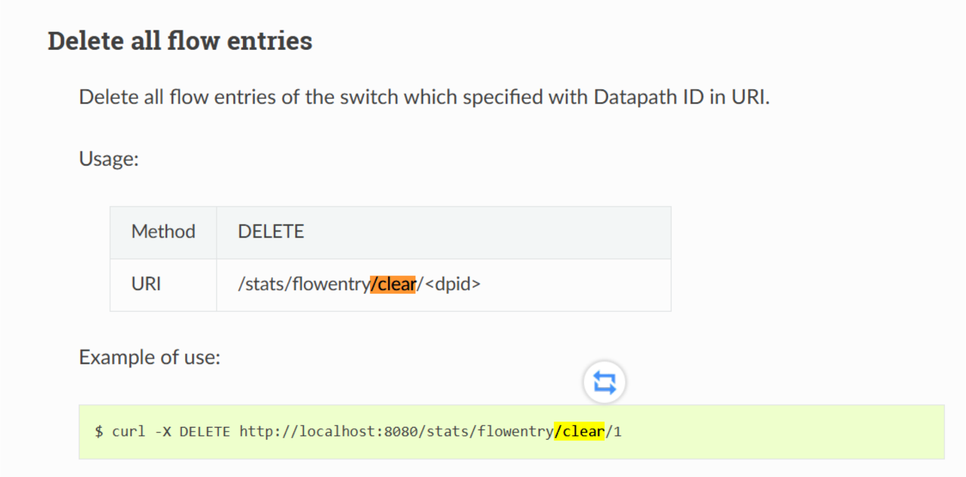
流表删除后网络恢复初始无连接状态:

```
mininet> h1 ping h2
Destination Host Unreachable
```

解决方案: 重新添加流表项或重启控制器。

6 结论

本文完整展示了 Mininet 与 Ryu 控制器的集成部署流程，详细说明了通过 RESTful API 操作 OpenFlow 流表的方法。通过添加/删除流表项实现了网络连通性的动态控制，为 SDN 网络研究和开发提供了实用参考。



Delete all flow entries

Delete all flow entries of the switch which specified with Datapath ID in URI.

Usage:

Method	DELETE
URI	/stats/flowentry/clear/<dpid>

Example of use:

```
$ curl -X DELETE http://localhost:8080/stats/flowentry/clear/1
```

Figure 12: 流表删除操作示例