P4实验

网络拓扑搭建

1. p4实验环境的网络可以通过 `p4utils` 提供的 API 使用 `Mininet` 框架来构建虚拟的网络环境， 可以有以下两种方式

   i. 通过python调用 `p4utils` 中的 API创建网络拓扑

```
from p4utils.mininetlib.network_API import NetworkAPI
net = NetworkAPI()
```

   常用的方法，详见 阅读network_API

```
net.addP4Switch(switch_name, cli_input = SW_COMMAND_FILE) #在虚拟网络中添加路由器
net.addHost(host_name) #在虚拟网络中添加主机
net.addLink(node1_name, node2_name) #为网络设备之间添加连接，可以是路由器与路由器之间，也可以是路
```

   创建网络拓扑

```
sudo python YOUR_TOPOLOGY_FILE
```

   ii. 将网络拓扑写入配置文件network.json，在命令行中使用p4run创建网络拓扑

```
sudo p4run --conf network.json
```

2. 通过可编程交换机
   - 主流芯片： `Intel Tofino` 芯片

     Intel® Tofino™ Series

     Meet the skyrocketing demand for bandwidth and data availability with the world's fastest P4-programmable Intelligent Fabric Processors.

     

     Intel® Tofino™ 2

     The next generation of programmable Ethernet switch, Intel® Tofino™ 2 is the best choice for meeting the needs of hyperscale data centers.

     Built with the same architecture as Intel® Tofino™, it's capable of delivering twice the bandwidth of its predecessor—up to 12.8 Tb/s.

     

     Intel® Tofino™

     Intel® Tofino™ is the world's first end-user programmable Ethernet switch. It is built using a Protocol Independent Switch Architecture (PISA) and is P4-programmable. Intel® Tofino™ switches are available in a selection of SKUs to suit different applications and needs.

   - 开发环境：Intel P4 Studio
   - 有关的open source code https://github.com/barefootnetworks/Open-Tofino/tree/master

控制层面

虚拟网络环境的控制层面一般有两种实现方式

1. command文件，如

   ```
   table_add table_name action name value1->value2
   ```

   然后于网络拓扑构建的文件中添加command文件

   ```
   "switches": {
      "s1": {
        "cli_input": "COMMAND_FILE",
        "program": "P4_FILE"
      }
    }
   ```

   或者在网络拓扑文件network.py中

   ```
   net.addP4Switch(switch_name, cli_input = COMMAND_FILE) #在拓扑中创建新路由器即绑定command文件
   ```

2. python调用控制层面API（更灵活）
   一般这部分需要实现一个controller类，以Thrift为例，需要包括以下基本代码, controller字段中控制层面的命令实现可参考 `/p4-utils/p4utils/mininetlib/network_API.py` 。下面代码块中的topology.json是虚拟网络构建成功后由 `p4utils` 相关工具自动生成的拓扑信息文件，无需手动编写。相关内容可参考 阅读topology.py.md

```
from p4utils.utils.helper import load_topo
from p4utils.utils.sswitch_thrift_API import *

class Controller(object){
   def __init__(self, sw_name){
      self.topo = load_topo('topology.json')
      self.sw_name = sw_name
      self.thrift_port = self.topo.get_thrift_port(sw_name)
      self.controller = SimpleSwitchThriftAPI(self.thrift_port)
   }
}
```

数据层面

p4 program运行在交换机的数据层面，在开发的过程中控制层面和数据层面
我们通过Sketch实现对于流的特征统计应该运行在数据层面

- 如何提取特征。
- 在哪里加载我们的模型？如果在控制层面加载模型，如何实现数据层面和控制层面的通信？如果数据层面可以加载模型，需要什么样的技术?

由于p4与协议无关，因此工作量取决于我们需要在l2,特别是l3除了基本的转发之外还实现什么样的网络路由转发功能(MPLS,ECMP等);

开发工具?

对于流的统计方案

如果我们把处理包\流的逻辑写，可以直接写在p4里面；但是如果我们需要动态处理，需要实时更改流表项，我们需要将包传输给controller进行处理，因为p4逻辑是写给交换机的，只能根据流表项来处理包，但是不能更改流表项，只有controller才有更改流表项的权限。

在p4程序内，也就是数据层面实现哈希和sketch，在controller获取sketch存储块（register[]）的内容并解码

- 如何在p4中实现复合结构sketch
- 更快速

将pkt传递给controller进行流的统计

- 在虚拟环境中如何传递pkt给controller
  - 将包clone pkt 并映射到cpu端口，监听该端口，把进入该端口的包交给controller进行处理。
  - 使用scapy实现端口监听，对pkt各层的拆分
- 涉及到pkt的clone与二次转发，速度可能会慢