P4lang介绍

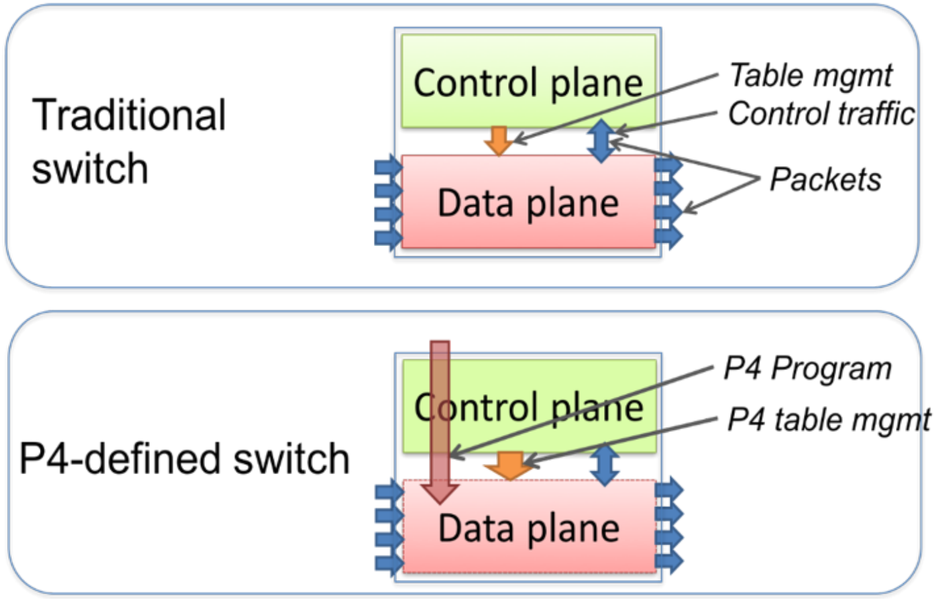
说明：方向键或空格查阅

简介

P4语言是一种用于对网络设备的数据平面进行编程，且与协议无关的数据包处理编程语言。
P4 最初设计是用于可编程的交换机中data plane的编程，目前已经扩展到了许多设备，
包括可编程网络接口卡、FPGA、软件交换机和硬件 ASIC，这些设备在P4中称为target。

P4可编程交换机与传统交换机的比较

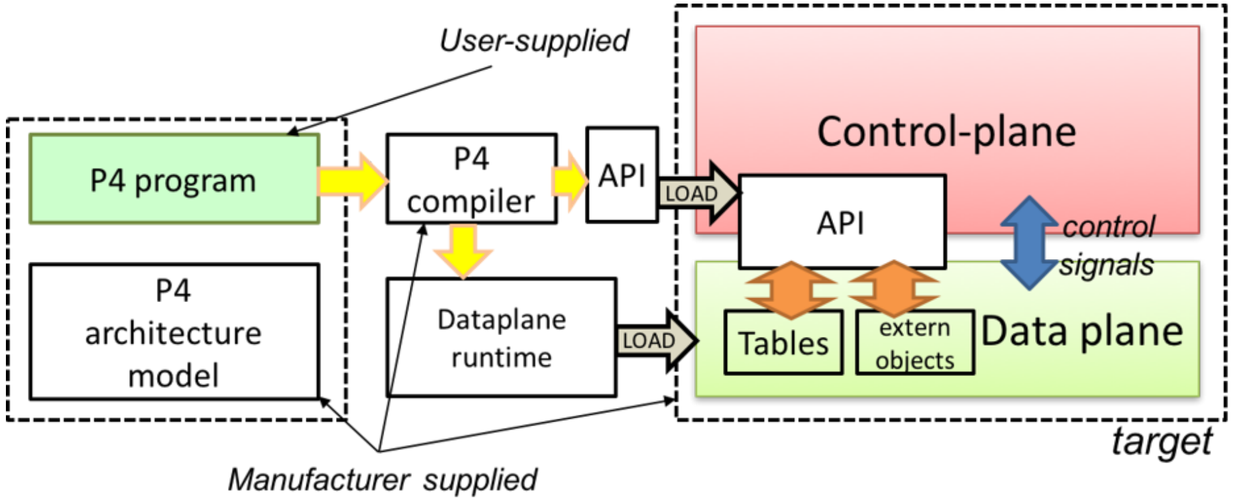
传统交换机中data plane的功能由制造商定义，
而P4可编程交换机的data plane的功能由P4程序定义。



P4 target编程的工作流

P4 architecture model可以看作是p4 program与target之间的约定，
开发者需要针对architecture完成p4 program的开发。

P4 compiler 由target制造商提供的，在编译p4 program时会产生两个artifacts：
描述转发逻辑的data plane配置,例如simple_switch的配置文件是个json文件。
用于control plane管理data plane对象状态的API，例如对某个表项进行添加、删除等操作。



安装

从软件源中安装

对于Ubuntu 20.04 、20.10 、 21.04 、21.10可以考虑使用软件源的方式安装。

```
. /etc/os-release

echo "deb http://download.opensuse.org/repositories/home:/p4lang/xUbuntu_${VERSION_ID}/ /" | tee /etc/apt/sources.lis

curl -L "http://download.opensuse.org/repositories/home:/p4lang/xUbuntu_${VERSION_ID}/Release.key" | apt-key add -

apt-get update
apt install -y libbpf p4lang-p4c p4lang-bmv2 p4lang-pi

```

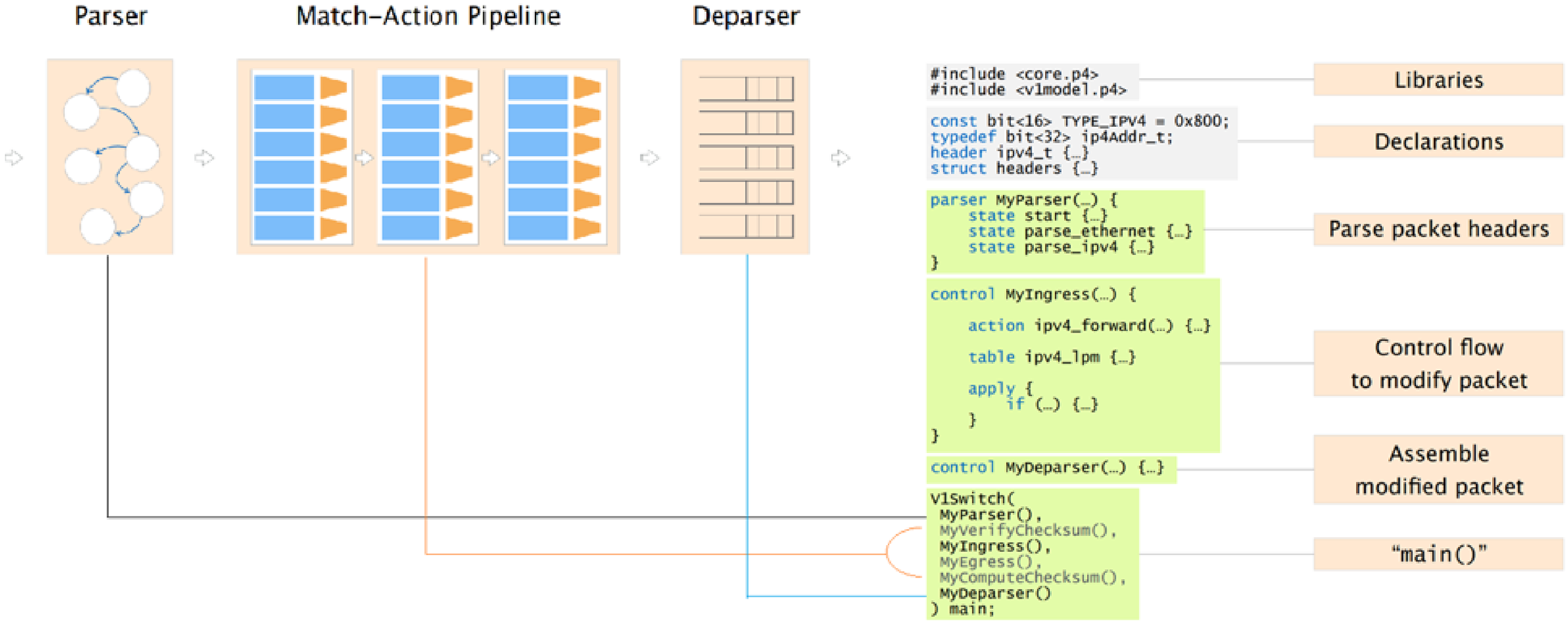
安装

源码安装 以Ubuntu16.04为例。

```
# 安装依赖
sudo apt-get install -y automake cmake libgmp-dev \
    libpcap-dev libboost-dev libboost-test-dev libboost-program-options-dev \
    libboost-system-dev libboost-filesystem-dev libboost-thread-dev \
    libevent-dev libtool flex bison pkg-config g++ libssl-dev \
    git   libgc-dev   libfl-dev  libboost-iostreams-dev wget\
    libboost-graph-dev llvm  python python-scapy python-ipaddr python-ply python3-pip \
    tcpdump libreadline-dev valgrind libtool-bin autoconf curl make unzip \
    build-essential  libpcre3-dev libavl-dev libev-dev libprotobuf-c-dev protobuf-c-compiler

pip3 install scapy ply

# Mininet安装
# 用于构建一个虚拟的网络拓扑。
git clone https://github.com/mininet/mininet
./mininet/util/install.sh -nwv

# protobuf安装
git clone --recursive -b v3.6.1 https://github.com/google/protobuf.git
cd protobuf
./autogen.sh
```

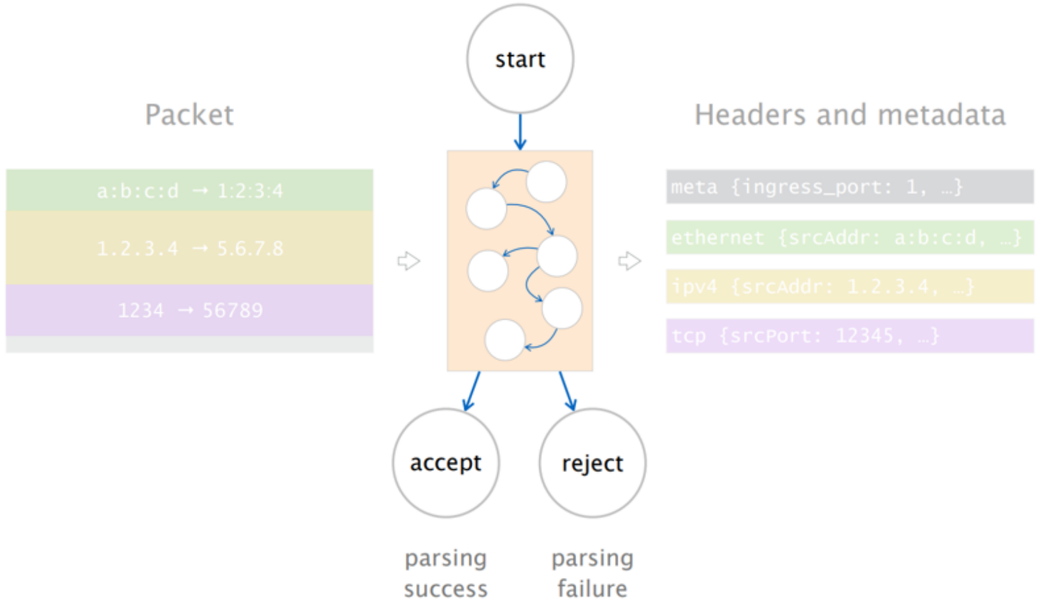
P4程序可以大致划分为3个部分

- Parser
- Match-Action Pipeline
- Deparser



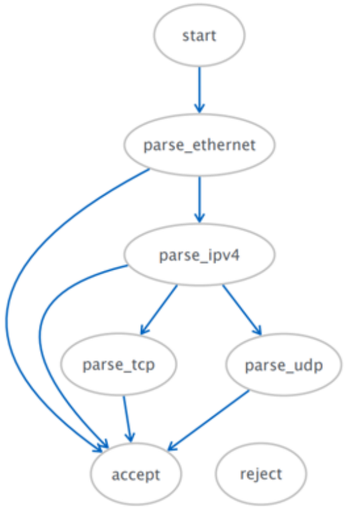
Parser

Parser 解析packet并获取所需的数据到headers and metadata。
Parser预定义了3种状态：start、accept和reject。



Parser实例

从start状态直接跳转到parse_ethernet。
parse_ethernet状态从packet提取ethernet，并判断是否值是否为0x800，若不满足则直接跳转到accept状态。
parse_ipv4中进一步提取，判断值是否对应tcp、udp的值并跳转到对应的状态。
...




```
parser MyParser(…) {
 state start {
  transition parse_ethernet;
 }
 state parse_ethernet {
  packet.extract(hdr.ethernet);
  transition select(hdr.ethernet.etherType) {
    0x800: parse_ipv4;
    default: accept;
  }
 }
 state parse_ipv4 {
  packet.extract(hdr.ipv4);
  transition select(hdr.ipv4.protocol) {
   6: parse_tcp;
   17: parse_udp;
   default: accept;
  }
 }
 state parse_tcp {
  packet.extract(hdr.tcp);
   transition accept;
 }
 state parse_udp {
  packet.extract(hdr.udp);
   transition accept;
 }
}
```

Match-Action Pipeline

指定表和相应的逻辑处理，例如改写mac地址，让packet发送到指定的port等。

- Action

  负责逻辑处理的实现，可以看作函数的声明

```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {

action ipv4_forward(macAddr_t dstAddr,
                    egressSpec_t port) {
 std_meta.egress_spec = port;
 hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
 hdr.ethernet.dstAddr = dstAddr;
 hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
}

 apply {
  ipv4_forward(0x123, 1);
 }
}
```

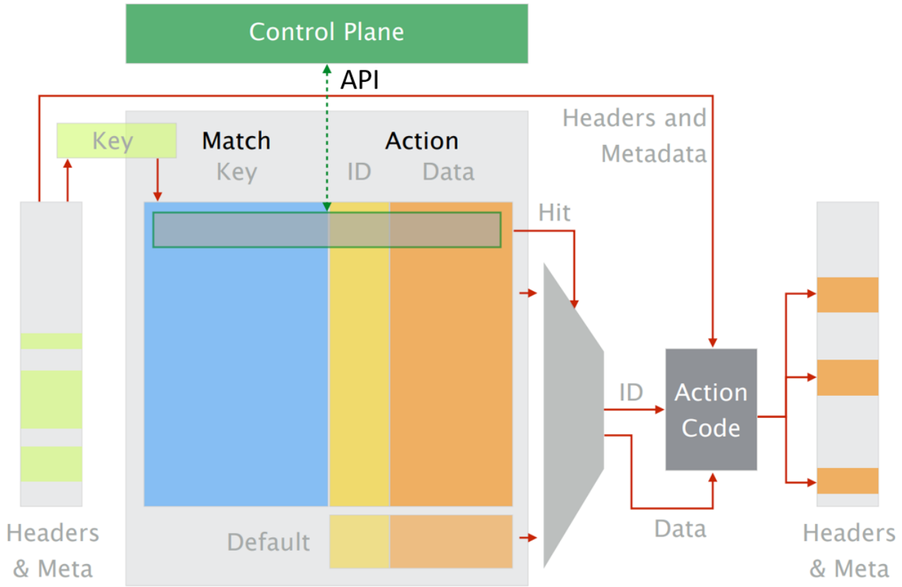
Control flow

Match-Action Pipeline

- Table

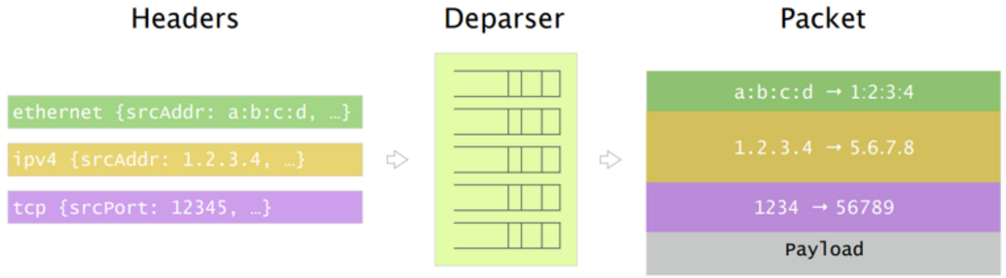
  将Parser解析所得的Headers & Metadata作为key在look-up table中查找，若命中则执行对应的Action，否则执行Default Action。

  Table对于data plane是只读的，但其条目可以由Control Plane修改。



Deparser

Deparser将修改后的headers重新组入到packet中并发送出去。




```
control MyDeparser(packet_out packet, in headers hdr) {
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.ipv4);
        packet.emit(hdr.tcp);
    }
}
```

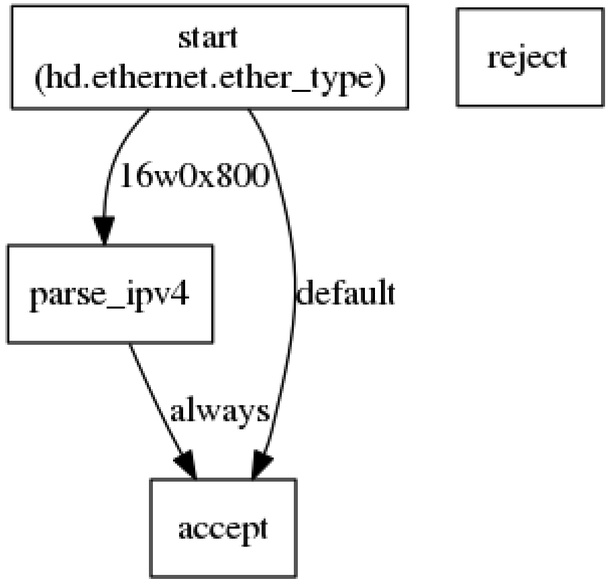
实例

创建example.p4文件

```
#include <core.p4>
#include <v1model.p4>

typedef bit<48> EthernetAddress;
typedef bit<32> IPv4Address;

header ethernet_t {
    EthernetAddress dst_addr;
    EthernetAddress src_addr;
    bit<16>         ether_type;
}

header ipv4_t {
    bit<4>      version;
    bit<4>      ihl;
    bit<8>      diffserv;
    bit<16>     total_len;
    bit<16>     identification;
    bit<3>      flags;
    bit<13>     frag_offset;
    bit<8>      ttl;
```

生成parser的表流图(可选)

```
# 解析成dot文件
p4c-graphs --std=p4-16 --target=bvm2 --arch=v1model  example.p4
# 生成可视化文件：pdf、svg、png等
dot -Tpng my_parser.dot -o parser.png
.
|-- example.p4
|-- example.p4i
|-- example.json
|-- my_compute_checksum.dot
|-- my_deparser.dot
|-- my_egress.dot
|-- my_ingress.dot
|-- my_parser.dot
|-- my_verify_checksum.dot
`-- parser.png
```



环境准备

准备网络接口

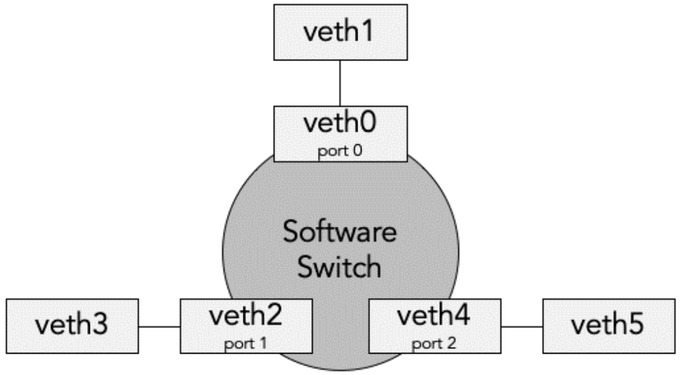
```
# First pair: veth0-veth1
ip link add name veth0 type veth peer name veth1
ip link set dev veth0 up
ip link set dev veth1 up
ip link set veth0 mtu 9500
ip link set veth1 mtu 9500
sysctl net.ipv6.conf.veth0.disable_ipv6=1
sysctl net.ipv6.conf.veth1.disable_ipv6=1

# Second pair: veth2-veth3
ip link add name veth2 type veth peer name veth3
ip link set dev veth2 up
ip link set dev veth3 up
ip link set veth2 mtu 9500
ip link set veth3 mtu 9500
sysctl net.ipv6.conf.veth2.disable_ipv6=1
sysctl net.ipv6.conf.veth3.disable_ipv6=1

# Third pair: veth4-veth5
ip link add name veth4 type veth peer name veth5
ip link set dev veth4 up
```

编译

```
p4c -v  --target bmv2 --arch v1model –std <program>.p4
#例 p4c -v  --target bmv2 --arch v1model –std example.p4
# -v, --debug                                                   verbose
# -b TARGET, --target                                           TARGET specify target device [bmv2,dpdk,ebpf]
# -a ARCH, --arch ARCH                                          specify target architecture [v1model,psa]
# --std {p4-14,p4_14,p4-16,p4_16}, -x {p4-14,p4_14,p4-16,p4_16} Treat subsequent input files as having type language
```

启动软件交换机

```
simple_switch --interface 0@<interface> --interface 1@ <interface> [... interface n@<interface>] <program>.json
#例 simple_switch --interface 0@veth0 --interface 1@veth2 --interface 2@veth4 example.json

##另外开窗口登录cli
simple_switch_CLI [--thrift-ip=<switch_ip>] [--thrift-port=<switch_port>] # switch_port 默认9090
#例 simple_switch_CLI
```



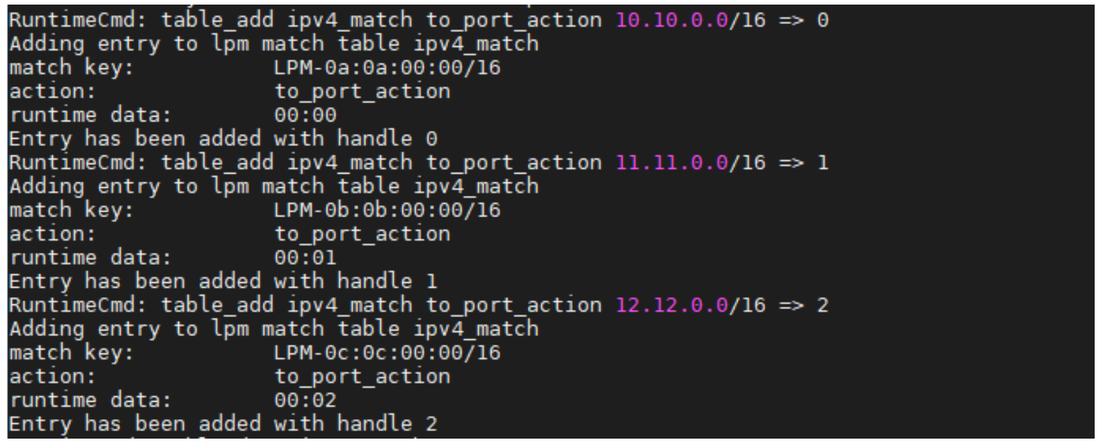
cli的使用示例

```
# 查看命令及帮助信息
RuntimeCmd: help
RuntimeCmd: help <command> # help show_tables
RuntimeCmd: show_tables

```

```
RuntimeCmd: help show_tables
List tables defined in the P4 program: show_tables
RuntimeCmd: show_tables
my_ingress.ipv4_match           [implementation=None, mk=ipv4.dst_addr(lpm, 32)]
tbl_example85                   [implementation=None, mk=]
RuntimeCmd:
```

添加路由

```
table_add ipv4_match to_port_action 10.10.0.0/16 => 0
table_add ipv4_match to_port_action 11.11.0.0/16 => 1
table_add ipv4_match to_port_action 12.12.0.0/16 => 2
```

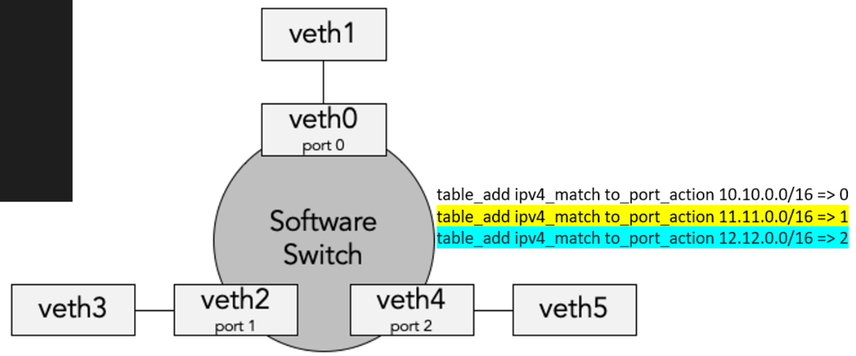


验证

- 对veth3抓包
- 使用scapy向veth1接口发送目的地址为11.11网段的数据包
- 检测veth3是否抓包是否有接收到，若有则表示数据包成功向指定port发送
- 同理，对veth5抓包，并使用scapy向veth1接口发送目的地址为12.12网段的数据包
- 检测veth5是否收到数据包

```
root@00b5cfd70bec:/# scapy
INFO: Can't import matplotlib. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: No IPv6 support in kernel
INFO: Can't import python-cryptography v1.7+. Disabled WEP decryption/encryption. (Dot11)
INFO: Can't import python-cryptography v1.7+. Disabled IPsec encryption/authentication.
WARNING: IPython not available. Using standard Python shell instead.
AutoCompletion, History are disabled.

                    aSPY//YASa
            apyyyyCY//////////YCa       |
           sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.5
 AYAsAYYYYYYYY///Ps              cY//S   |
        pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
        SPPPP///a          pP///AC//Y   |
             A//A            cyP////C   | Have fun!
             p///Ac            sC///a   |
             P////YCpc           A//A   | To craft a packet, you have to be a
      scccccp///pSP///p          p//Y   | packet, and learn how to swim in
     sY/////////y  caa           S//P   | the wires and in the waves.
      cayCyayP//Ya             pY/Ya    |        -- Jean-Claude Van Damme
       sY/PsY////YCc          aC//Yp    |
        sc  sccaCY//PCypaapyCP//YSs
                 spCPY//////YPSps
                      ccaacs

>>> sendp(Ether()/IP(dst="11.11.1.1")/UDP(), iface="veth1")
.
Sent 1 packets.
>>>
>>> sendp(Ether()/IP(dst="12.12.1.1")/UDP(), iface="veth1")
.
Sent 1 packets.
>>>
```

```
root@00b5cfd70bec:/# tcpdump -n -i veth3
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth3, link-type EN10MB (Ethernet), capture size 262144 bytes
14:29:23.587593 IP 172.17.0.2.53 > 11.11.1.1.53: domain [length 0 < 12] (invalid)
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
root@00b5cfd70bec:/#
root@00b5cfd70bec:/#
root@00b5cfd70bec:/#
root@00b5cfd70bec:/#
root@00b5cfd70bec:/#
root@00b5cfd70bec:/# tcpdump -n -i veth5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on veth5, link-type EN10MB (Ethernet), capture size 262144 bytes
14:36:35.422180 IP 172.17.0.2.53 > 12.12.1.1.53: domain [length 0 < 12] (invalid)
```



参考

https://github.com/p4lang/PI

https://github.com/p4lang/p4c

https://github.com/nsg-ethz/p4-learning

https://github.com/p4lang/behavioral-model

https://p4.org/p4-spec/docs/P4-16-v1.2.2.html

https://build.opensuse.org/project/show/home:p4lang

https://github.com/p4lang/tutorials/blob/master/vm/user-bootstrap.sh

https://github.com/protocolbuffers/protobuf/blob/master/src/README.md

https://opennetworking.org/news-and-events/blog/getting-started-with-p4