数据平面的可编程时代

P4应用与实战

江苏省未来网络创新研究院　杨帅

提纲

- P4编程快速开始
- P4与ONOS
- P4与heavy fitter
- P4与INT vSwitch
- P4语言编程

P4项目源码目录结构

p4c-bm

- 后端编译器，可将高级语言或高级语言中间表示转为JSON格式或PD格式的配置文件

ptf

- Python测试框架，基于unittest框架实现，该框架中的大部分代码从floodlight项目中的OFTest框架移植而来

p4-hlir

- 前端编译器，将高级抽象语言转化成高级语言中间表示

P4ofagent

- OpenFlow协议插件，目前实现的功能有限

p4c-behavior

- 第一代bm的编译器，现已基本废弃

ntf

- 网络测试框架，集成了mininet和docker，包含较多bmv2应用测试脚本

behavioural—model

- 模拟P4数据平面的用户态软件交换机bmv2，识别JSON格式配置文件

switch

- Switch示例，基本完成交换机的绝大部分功能

p4factory

- 快速开始，内含6个可快速启动的项目basic_routing、copy_to_cpu、l2_switch、sai_p4、simple_router、switch

P4-build

- 需要手动生成的基础设施库，为执行P4程序编译、安装PD库

scapy-vxlan

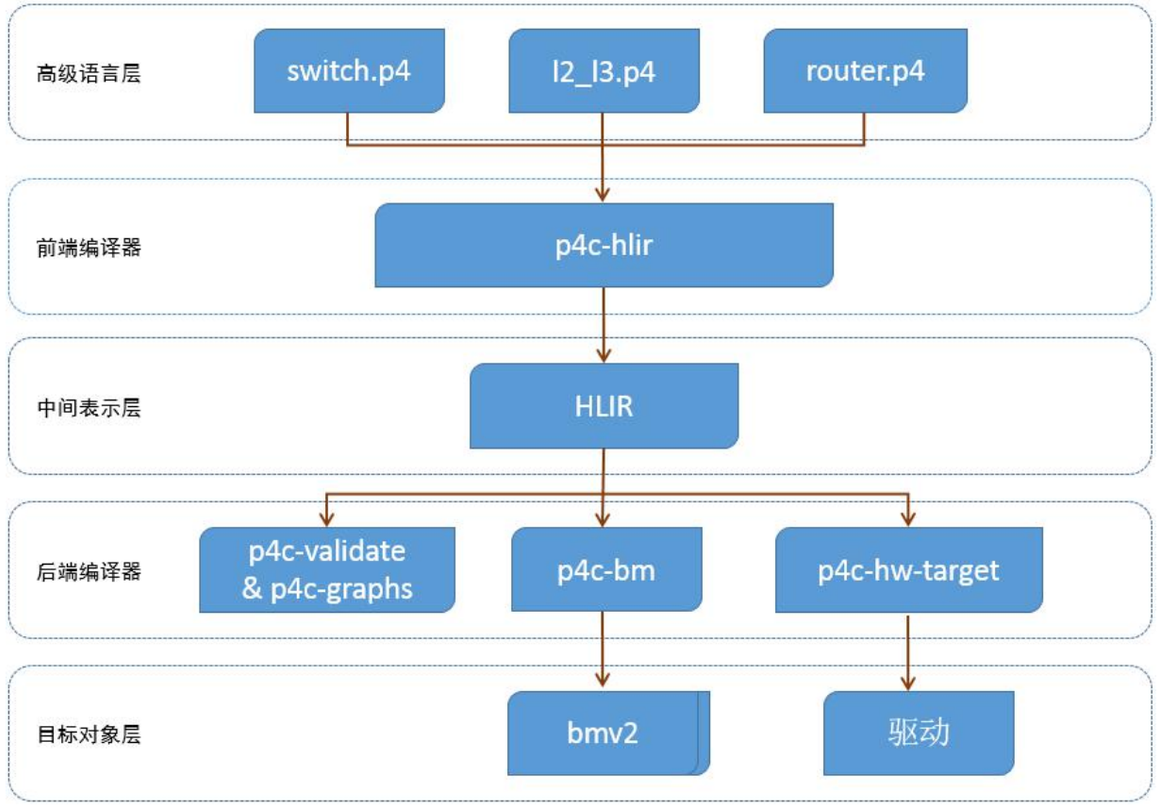
- 扩展VXLAN和ERSPAN-like协议包头处理

tutorials

- 示例教程，内含8个基础示例教程：cpoy_to_cpu、meter、TLV_parsing、register、counter、action_profile、resubmit、simple_nat

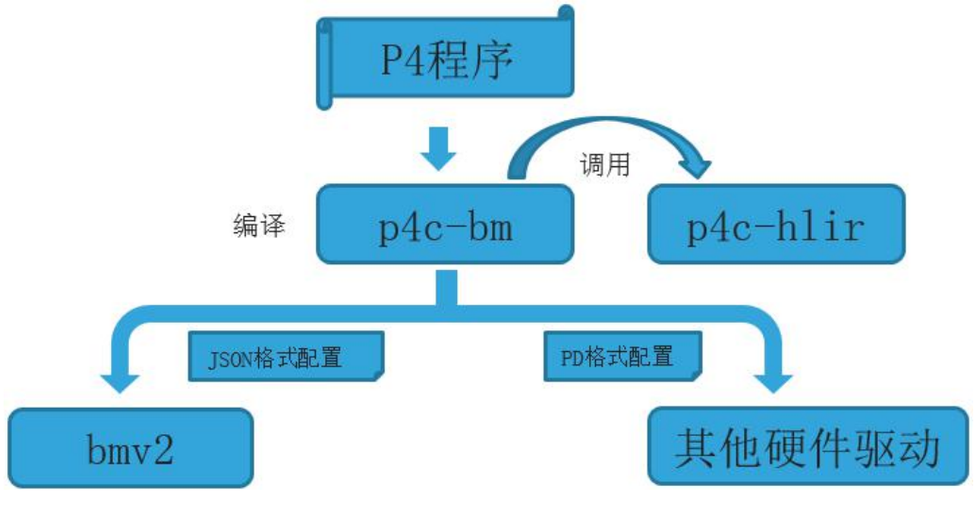
P4项目架构

- 高级语言层：高度抽象的P4语言编写的程序
- 前端编译器：对高级语言进行与目标无关的语义分析并生成中间表示
- 中间表示层：高级语言中间表示，可转换成多种其他语言
- 后端编译器：将中间表示转换为目标平台机器码
- 目标对象层：受控制硬件/软件设备



BMV2

P4项目中的behavioral model模块，是模拟P4数据平面的用户态软件交换机，使用C++语言编写，简称bmv2。P4程序首先经过p4c-bm模块编译成JSON格式的配置文件，然后将配置文件载入到bmv2，转化成能实现交换机功能的数据结构。



快速开始（P4FACTORY）

p4factory提供整套用以运行和开发P4程序环境的代码，帮助用户快速开发P4程序。

basic_routing	copy_to_cpu	l2_switch
• 基本路由功能，包括IPv4_lpm、mac重写、ipv4_fib、下一跳等功能	• 将以太网帧头部发送到CPU接口	• 基本交换功能，包括端口转发、mac地址学习、多播等功能

simple_router	switch	sai_p4
• 功能与basic_routing类似	• 基于bmv2的交换机实现，已支持大部分协议，默认使用switch_bmv2.json配置交换机，建立一个8端口的交换机	• 提供设备无关的控制转发元件的接口

IPv4 and IPv6 routing
- Unicast
- *Unicast RPF*
  - Strict and Loose
- Multicast
  - PIM-SM/DM & *PIM-Bidir*

L2 switching
- Learning
- STP state
- VLAN Translation
  - *Private VLANs*

Load balancing
- ECMP and LAG
- Resilient Hashing

Tunneling
- IPv4 & IPv6 Routing & Switching
- VXLAN, NVGRE, GENEVE & GRE

MPLS
- LSR, LER
- IPv4/v6 routing (L3VPN)
- L2 switching (EoMPLS, VPLS)

ACL/QOS
- MAC ACL,
- IPv4/v6 ACL/RACL,
- QoS ACL
- System ACL
- PBR (Policy based routing)
- CoPP (Control plane policing)

NAT
- Unicast and Multicast

Security Features
- Storm Control, IP Source Guard

Mirroring
- Ingress and Egress Mirroring

Counters
- Route Table Entry Counters

Protocol Offload
- BFD, OAM

Multi-chip Fabric Support
- Forwarding, QOS

FCoE
- FCF and NPV modes

- 系统

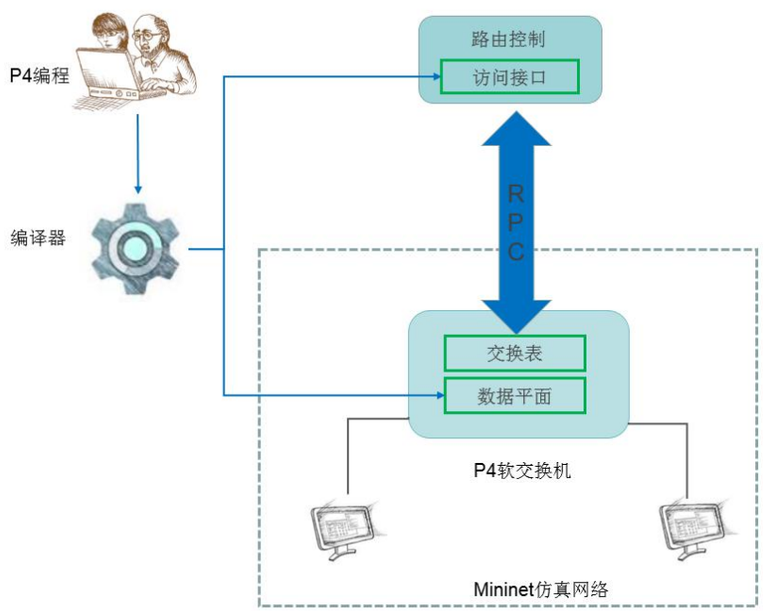
  推荐系统：Ubuntu14.04+

- 安装编译

```
$./install_deps.sh

$./autogen.sh

$./configure

$cd
targets/basic_routing

$make bm
```

- 运行

```
sudo ./behavioural-
model
```



提纲

P4与ONOS

P4+ONOS架构

特性

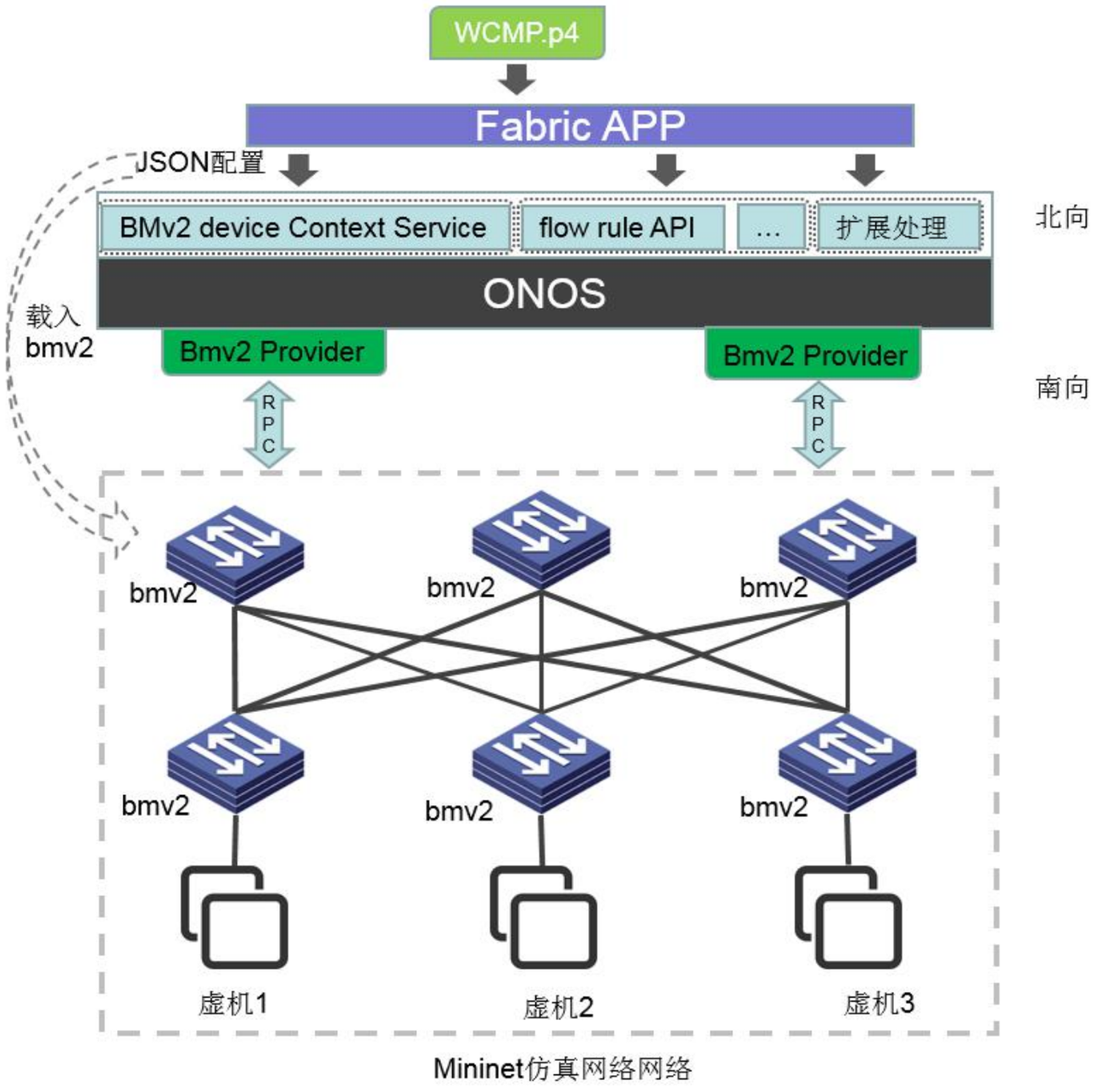
北向接口	南向接口
• 北向核心接口 • Bmv2 service接口 • 扩展处理接口	• Bmv2 Provider

控制器主动连接bmv2
• RPC调用

ONOS1.6+P4支持特性

- 设备发现：连接/断开事件
- 支持JSON配置转换
- 支持packet-in和packet-out动作
- 匹配-动作表填充（通过流规则，流目标或意图）
- 端口统计数据收集
- 流量统计数据收集

架构图

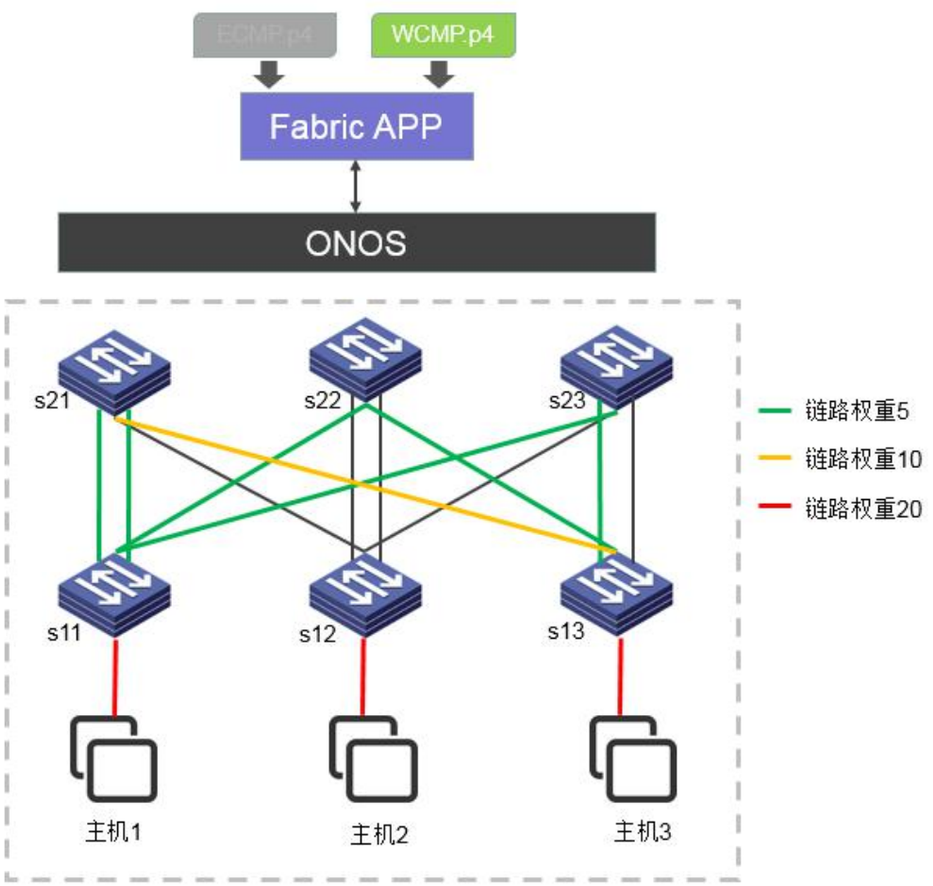


Mininet仿真网络网络

ONOS1.6中附带了两个P4示例程序：ECMP.p4和WCMP.p4。

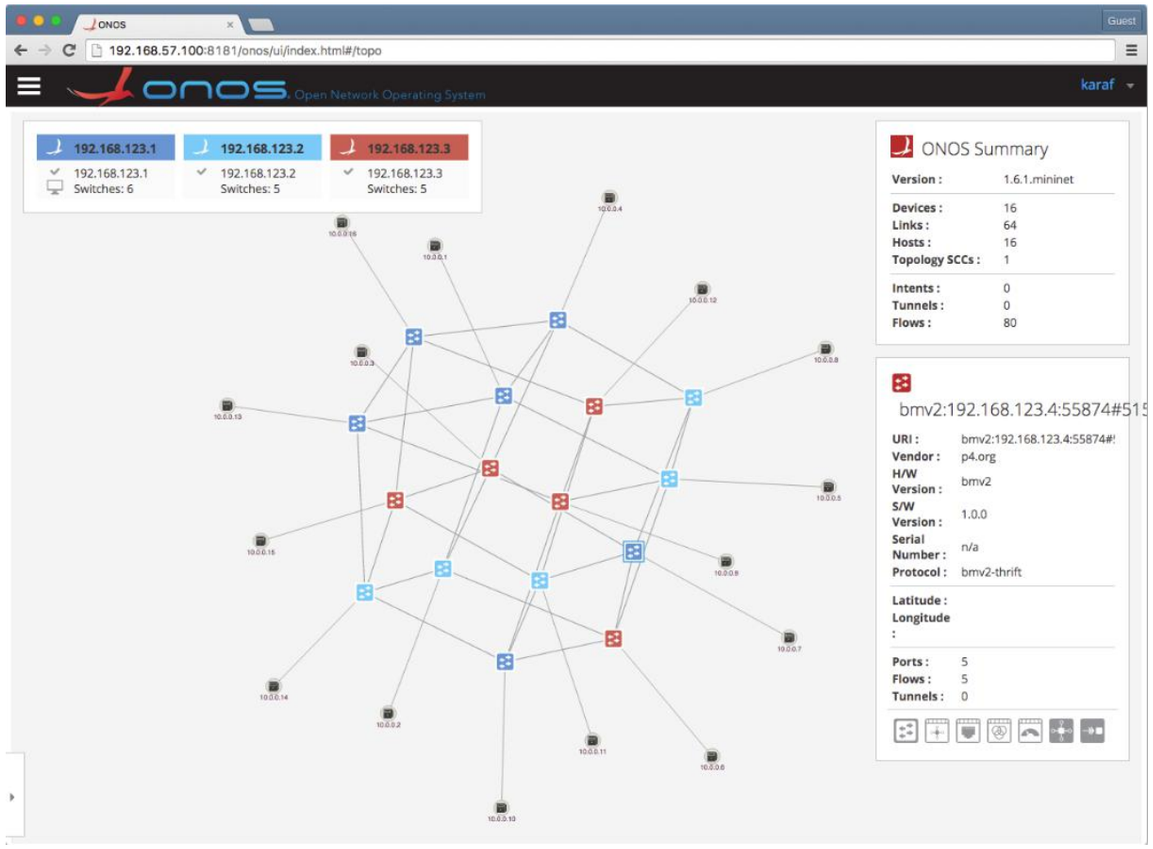
用户只需编译安装ONOS并激活*BMv2 Drivers*应用，然后使用onos提供的自定义拓扑脚本创建mininet仿真网络。

WCMP示例网络结构图



Mininet仿真网络网络

网络拓扑

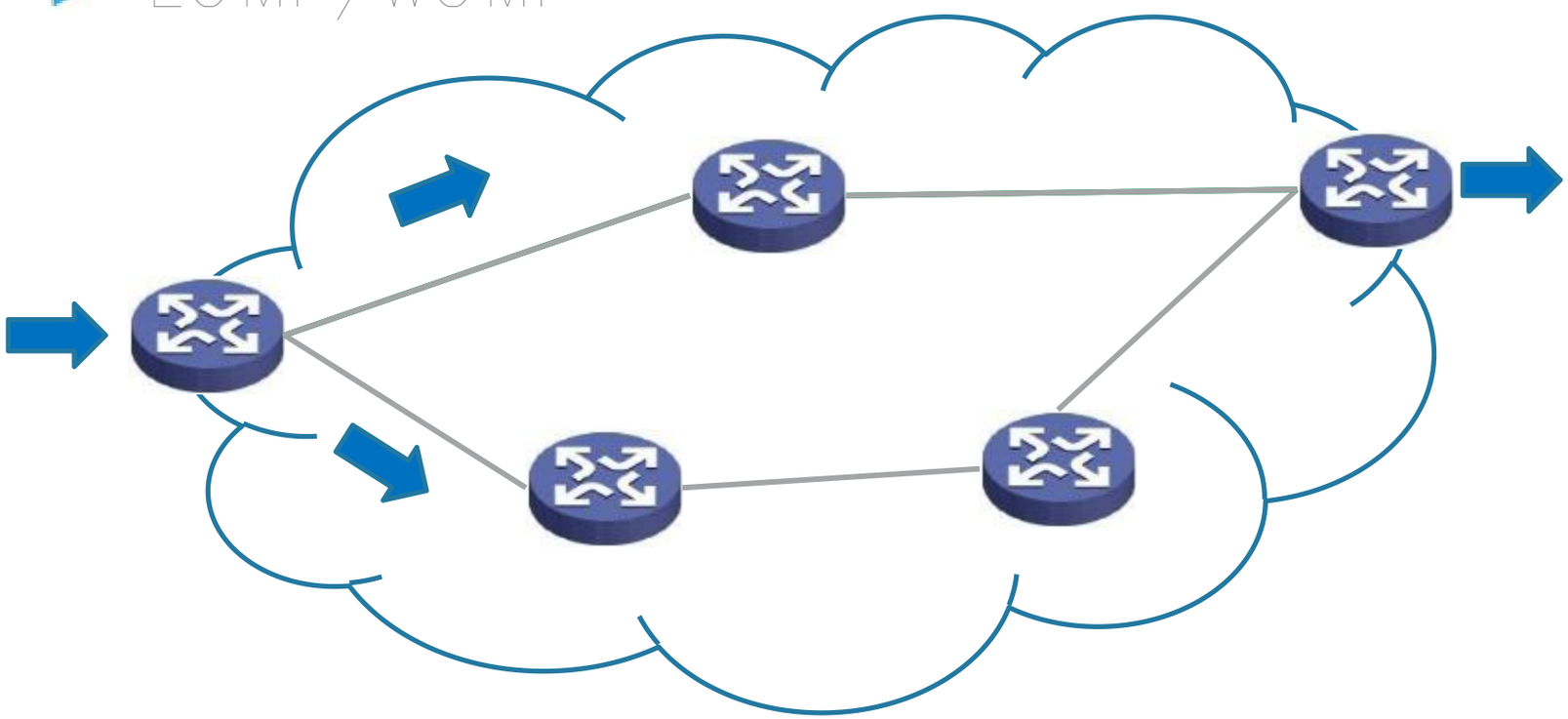


FLOWLET和ECMP/WCMP

- FLOWLET

基于包的分流	基于流的分流
• 分流比例准确 • TCP包乱序 • 容易动态调整比例	• 无法预测流大小 • 不会乱序 • 较难动态调整比例

- ECMP/WCMP

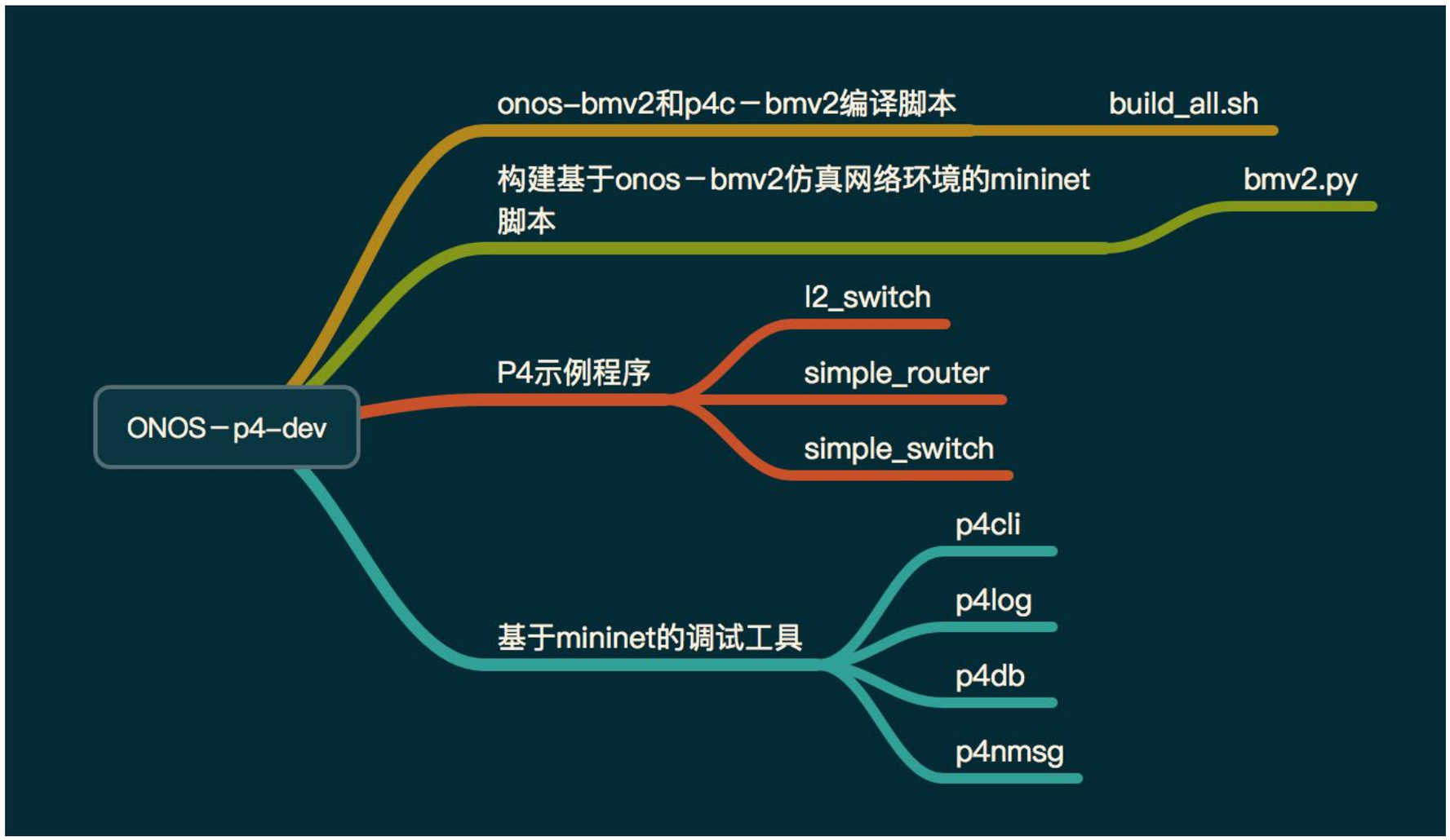


- 传统的路由协议都是采用单路径路由的方式
- 到达一个目的地有多条相同度量值的路由项(路由路径)
- 将所有链路权重视为相等，在链路上等比例传递流量
- 按照预设权重，按照不同比例在链路上传递流量

开发环境

onos-p4-dev

项目源码通过github下载，ONOS提供了整套基于bmv2的P4语言编程的开发、测试、运行环境



ECMP程序

ECMP元数据定义：

```
header_type ecmp_metadata_t {
    fields {
        groupId : 16;
        selector : 16;
    }
}
```

定义进行哈希计算的字段、指定哈希算法：

```
field_list ecmp_hash_fields {
    ipv4.srcAddr;
    ipv4.dstAddr;
    ipv4.protocol;
    tcp.srcPort;
    tcp.dstPort;
    udp.srcPort;
    udp.dstPort;
}

field_list_calculation ecmp_hash {
    input {
        ecmp_hash_fields;
    }
    algorithm : bmv2_hash;
    output_width : 64;
}
```

定义匹配表：

```
table table0 {
    reads {
        standard_metadata.ingress_port : ternary;
        ethernet.dstAddr : ternary;
        ethernet.srcAddr : ternary;
        ethernet.etherType : ternary;
    }
    actions {
        set_egress_port;
        ecmp_group;
        send_to_cpu;
        _drop;
    }
    support_timeout: true;
}

table ecmp_group_table {
    reads {
        ecmp_metadata.groupId : exact;
        ecmp_metadata.selector : exact;
    }
    actions {
        set_egress_port;
    }
}
```

指定匹配方式为三元匹配

指定匹配方式为精确匹配

数据包处理流程：

```
control ingress {
    apply(table0) {
        ecmp_group {
            apply(ecmp_group_table);
        }
    }
    process_port_counters();
}
```

提纲

P4与HEAVY HITTER

WHAT?

Heavy Hitter可以简单地定义为发送异常大的流量的流量源

WHY?

有针对性的进行流量调度优化，防止网络拥塞

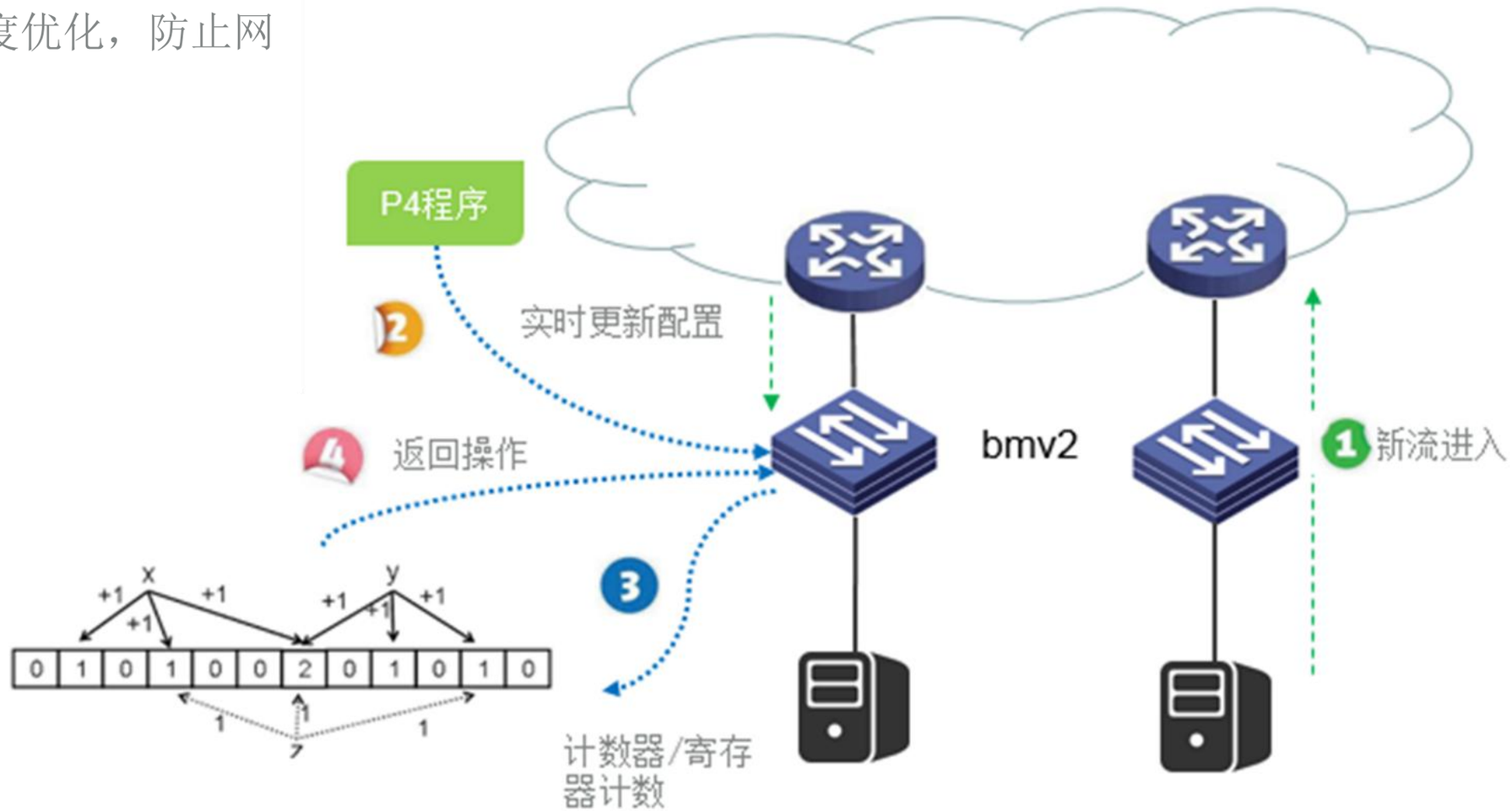
HOW?

1.基于IP地址分类

2.基于应用会话分类

基于P4实现

基于P4计数器(counter)和寄存器(register)的实现。



定义IP包计数器

count可以在流水线中任意的table中调用

```
counter ip_src_counter {
    type: packets;
    static: count_table;
    instance_count: 1024;
}

action count_action(idx) {
    count(ip_src_counter, idx);
}

table count_table {
    reads {
        ipv4.srcAddr : lpm;
    }
    actions {
        count_action;
        _drop;
    }
    size : 1024;
}
```

进行对源IP进行，lpm匹配。匹配成功后计数器+1

转发到流水线出口

出口流水线只对出端口进行精确匹配并转发

```
control egress {
    apply(send_frame);
}
```

数据包在流水线出口处重写mac地址

```
table send_frame {
    reads {
        standard_metadata.egress_port: exact;
    }
    actions {
        rewrite_mac;
        _drop;
    }
    size: 256;
}
```

在流水线入口处执行转发和计数

```
control ingress {
    apply(count_table);
    apply(ipv4_lpm);
    apply(forward);
}

control egress {
    apply(send_frame);
}
```

入口行精确匹配并转发

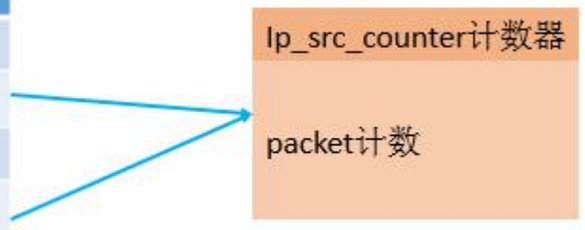
```
counter ip_src_counter {
    type: packets;
    static: count_table;
    instance_count: 1024;
}

action count_action(idx) {
    count(ip_src_counter, idx);
}

table count_table {
    reads {
        ipv4.srcAddr : lpm;
    }
    actions {
        count_action;
        _drop;
    }
    size : 1024;
}
```

计算匹配成功的分组包数量，该计数器的实例化数量上线为1024

任何表(table)可以通过地址+idx引用该counter

Action只能在table中调用，否则编译器报错

匹配字段	动作	数据
ABCD_0123		idx
Mactched entry	count_action	idx A
		idx
Mactched entry	count_action	idx A
		idx



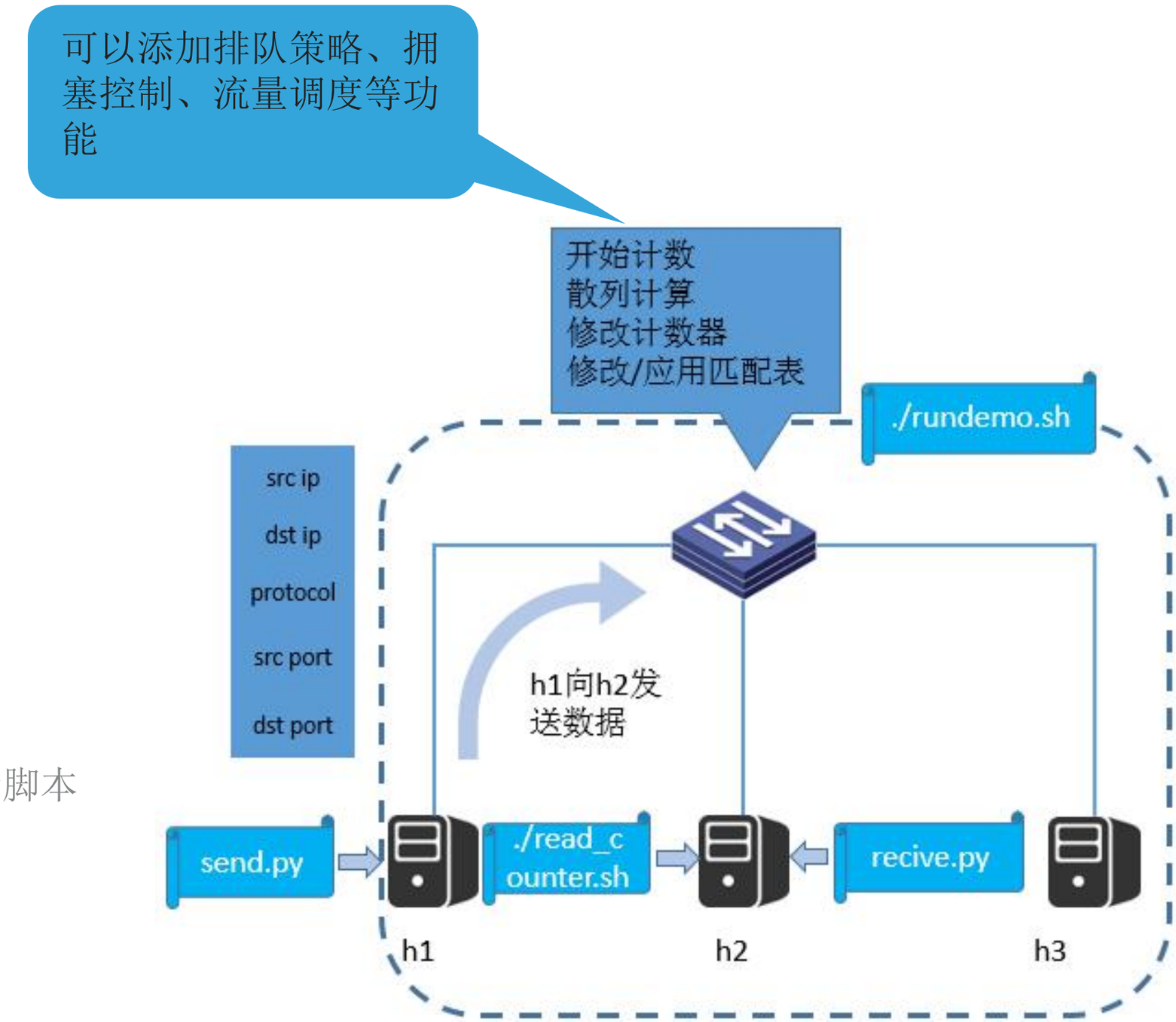
实验流程

运行环境仿真脚本

运行收脚本

运行发包脚本

运行计数器显示脚本



提纲

P4与INT
VSWITCH

基于P4网络监测

传统网络监测

- 客户端/服务器模式

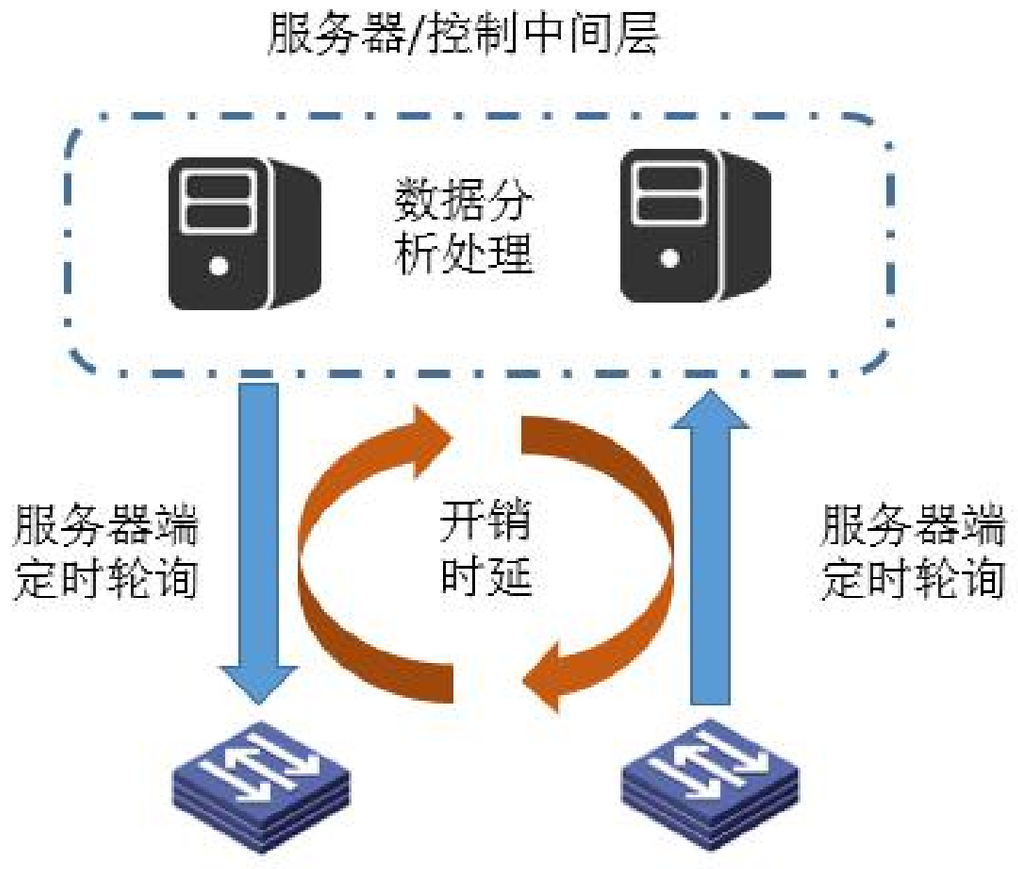
缺点

实时性

- 涉及CPU和控制层面
- 无法监控快速变化的网络状态

端到端网络状态

- 无法将流的实际路径和元素状态关联



In-band模式

In-band模式网络状态监测机制

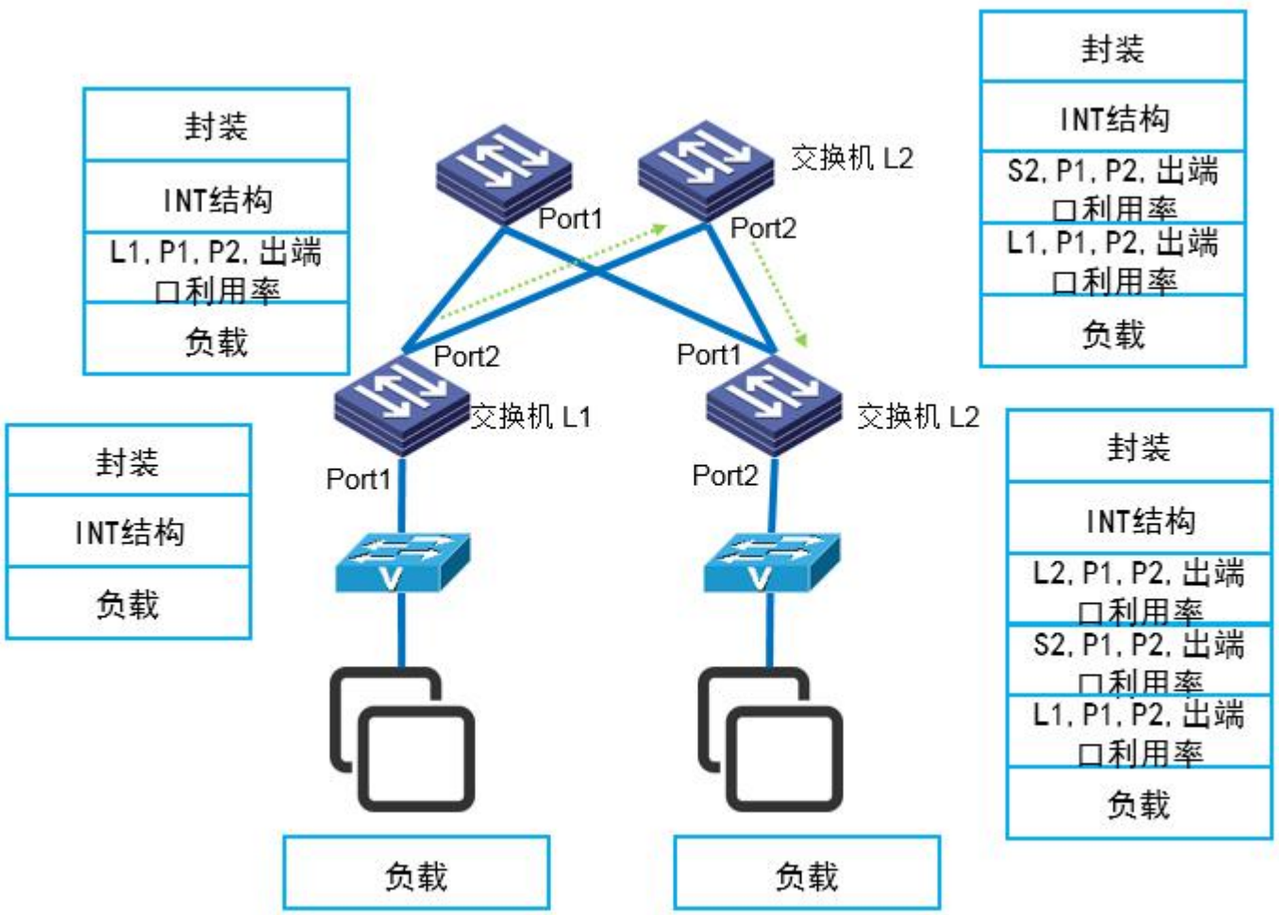
- 目的交换机将收集的数据发送到本地CPU本地设备上处理
- 目的交换机将数据导出到远程服务器上的网管系统或大数据分析工具
- 目的交换机将数据反馈到源交换机，以作为源交换机处理其他数据参考

网络状态元素

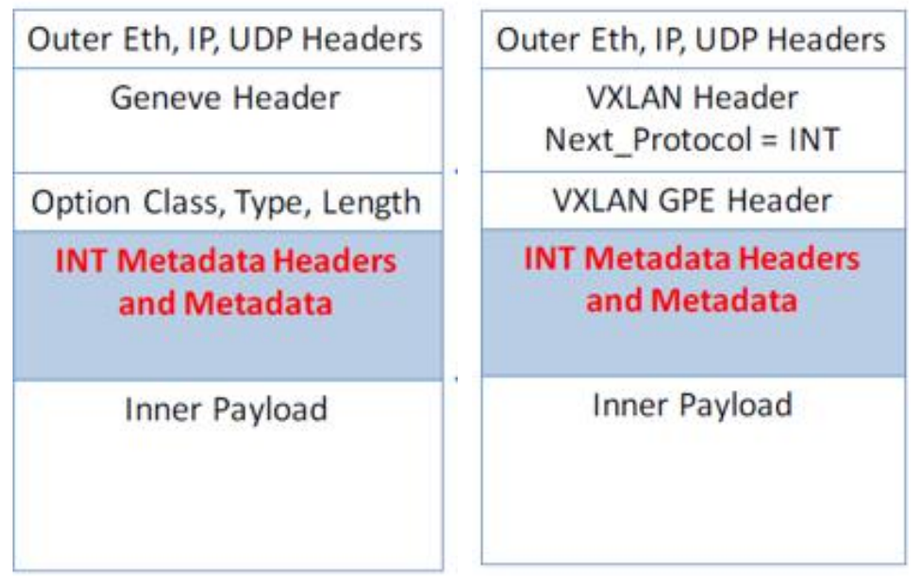
- 交换机ID+入端口ID+出端口ID—确定交换机端到端之间的不同路径。而传统的基于IP的路由追踪监测方式只能监控3层路径，无法区分Port-channel里的多条链路。
- 链路利用率—交换机端到端的不同路径的链路利用率既可以用于基础的监控，也可以用于选择新流量的路径选择，而不是盲目地将流量导向任意等价路径上。
- 延迟——交换机端到端的不同路径的链路延迟既可以用于基础的监控，也可以用于时间敏感类业务的智能路由。

P4网络监测场景实现

应用场景



INT 包头封装结构

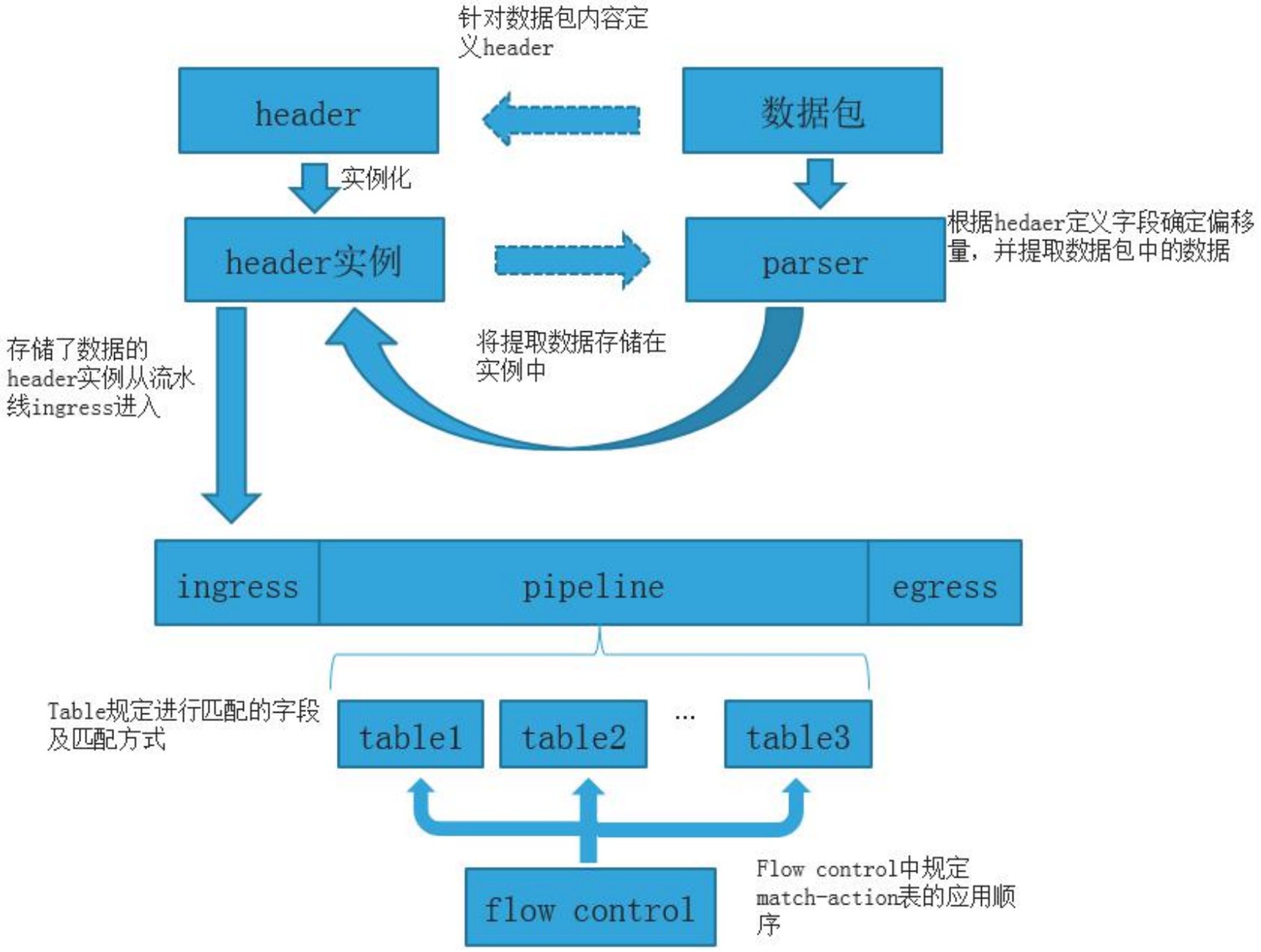


提纲

P4语言编程

如何编写P4程序？

- P4定义数据平面流程



header:

```
header_type ecmp_metadata_t {
    fields {
        groupId : 16;
        selector : 16;
    }
}
```

parser:

```
parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        ETHERTYPE_IPV4 : parse_ipv4;
        default: ingress;
    }
}
```

table:

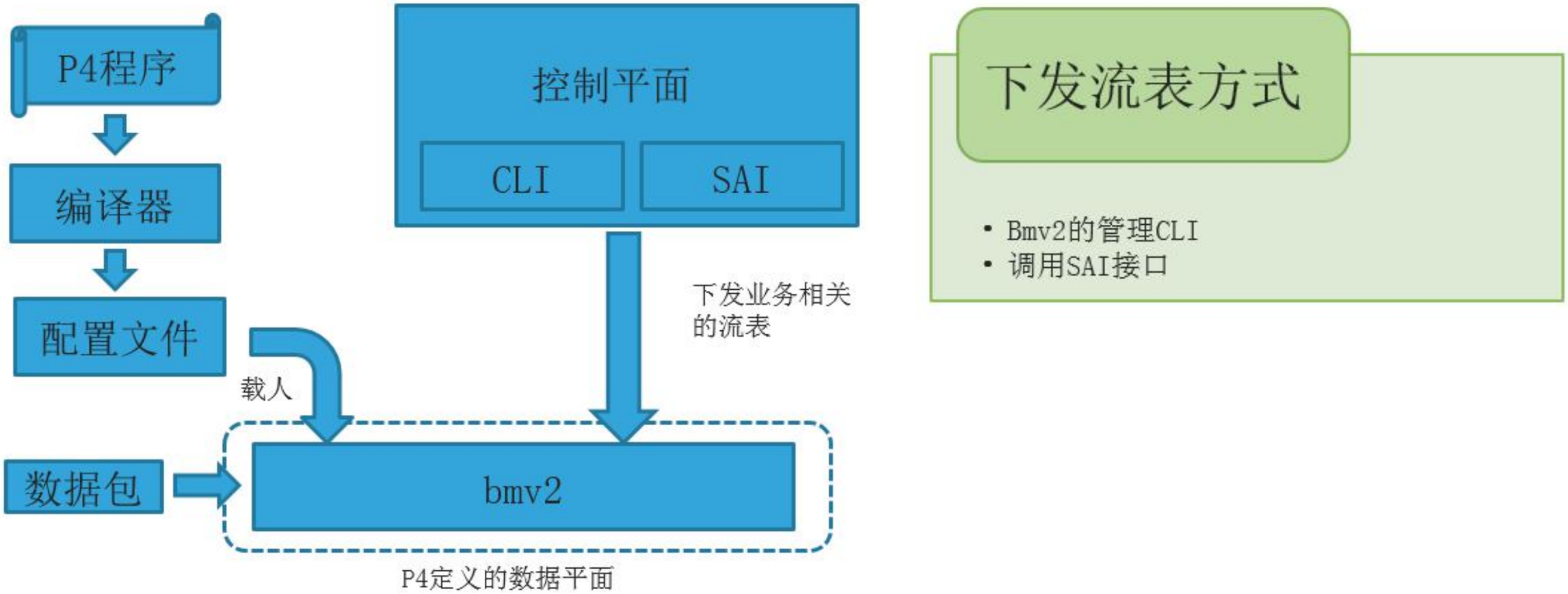
```
table send_frame {
    reads {
        standard_metadata.egress_port: exact;
    }
    actions {
        rewrite_mac;
        _drop;
    }
    size: 256;
}
```

control:

```
control ingress {
    apply(count_table);
    apply(ipv4_lpm);
    apply(forward);
}

control egress {
    apply(send_frame);
}
```

基于P4的控制平面

- P4中的控制平面与数据平面

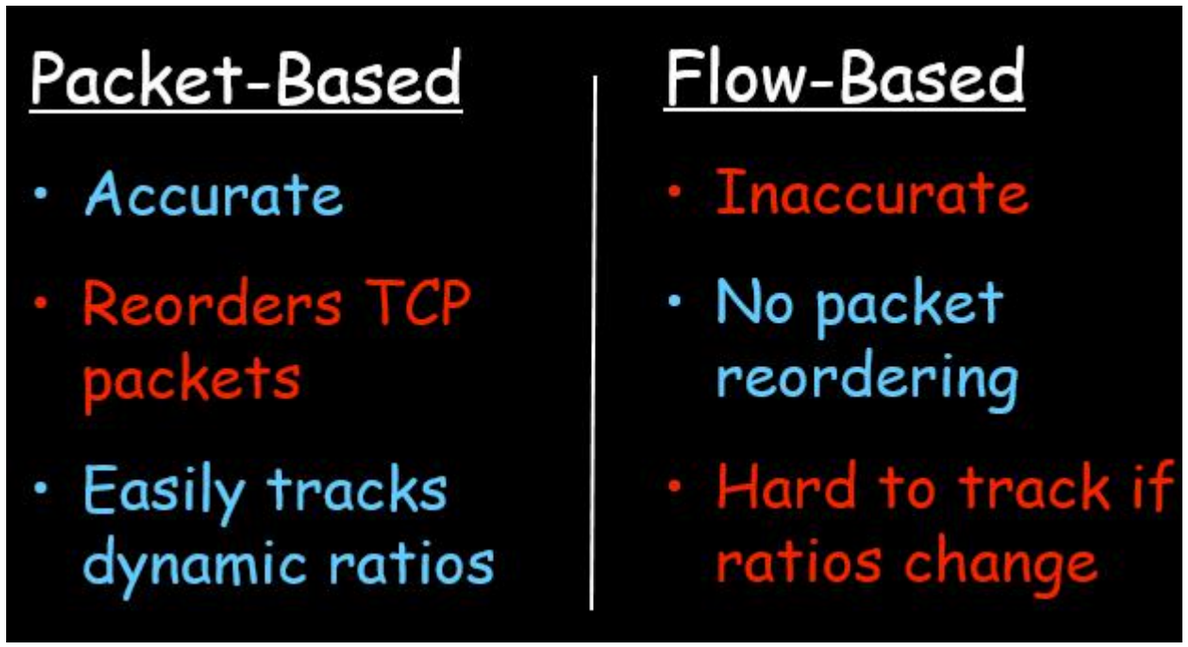


- Bash脚本中调用CLI

```
python ../../cli/pd_cli.py -p l2_switch -i p4_pd_rpc.l2_switch -s $PWD/tests/pd_thrift:$PWD/../../testutils -m "set_default_action dmac broadc
ast" -c localhost:22222
python ../../cli/pd_cli.py -p l2_switch -i p4_pd_rpc.l2_switch -s $PWD/tests/pd_thrift:$PWD/../../testutils -m "set_default_action mcast_src_p
runing _nop" -c localhost:22222
python ../../cli/pd_cli.py -p l2_switch -i p4_pd_rpc.l2_switch -s $PWD/tests/pd_thrift:$PWD/../../testutils -m "add_entry mcast_src_pruning 5
_drop" -c localhost:22222
mgrphdl=`python ../../cli/pd_cli.py -p l2_switch -i p4_pd_rpc.l2_switch -s $PWD/tests/pd_thrift:$PWD/../../testutils -m "mc_mgrp_create 1" -c
localhost:22222 | awk '{print $NF;}'`
echo $mgrphdl > mgrp.hdl
l1hdl=`python ../../cli/pd_cli.py -p l2_switch -i p4_pd_rpc.l2_switch -s $PWD/tests/pd_thrift:$PWD/../../testutils -m "mc_node_create 0 30 -1"
 -c localhost:22222 | awk '{print $NF;}'`
echo $l1hdl > l1.hdl
python ../../cli/pd_cli.py -p l2_switch -i p4_pd_rpc.l2_switch -s $PWD/tests/pd_thrift:$PWD/../../testutils -m "mc_associate_node $mgrphdl $l1
hdl" -c localhost:22222
```

谢谢观看！

江苏省未来网络创新研究院-杨帅@SDNLAB
yangshuai@sdnlab.com



基于包的分流

- 分流比例准确
- TCP包乱序
- 容易动态调整比例

基于流的分流

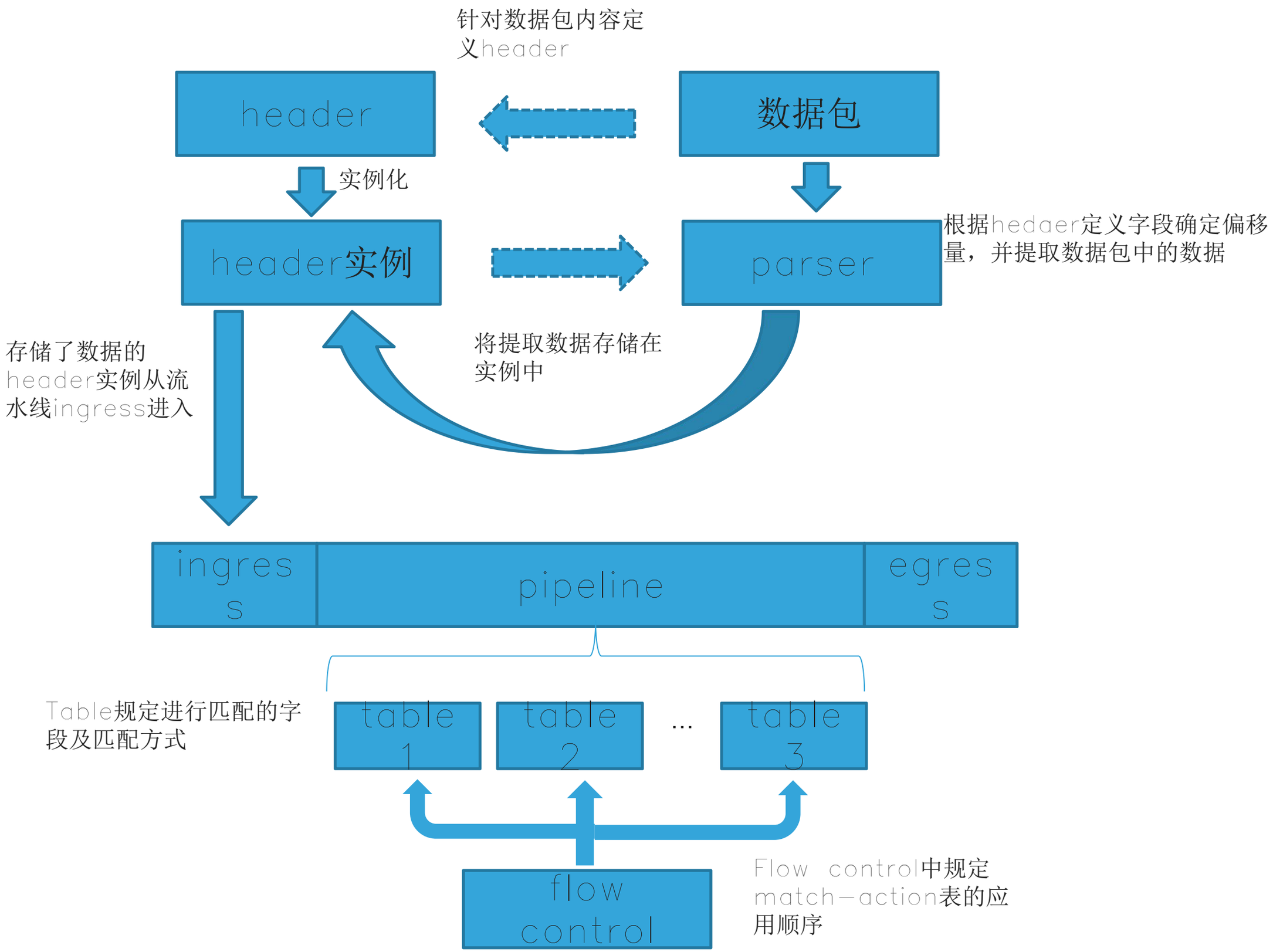
- 无法预测流大小
- 不会乱序
- 较难动态调整比例

In-band模式网络状态监测机制

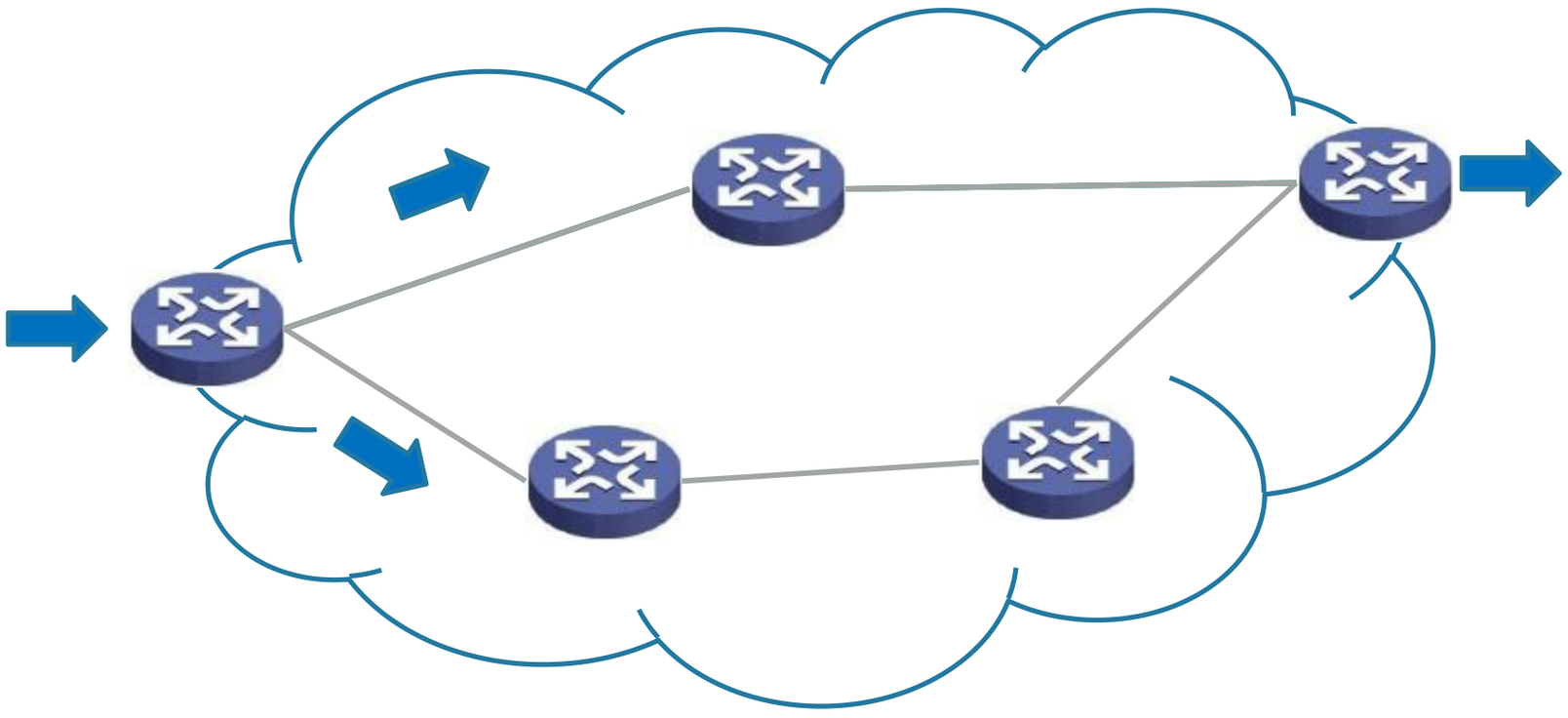
- 目的交换机将收集的数据发送到本地CPU本地设备上处理
- 目的交换机将数据导出到远程服务器上的网管系统或大数据分析工具
- 目的交换机将数据反馈到源交换机，以作为源交换机处理其他数据参考

网络状态元素

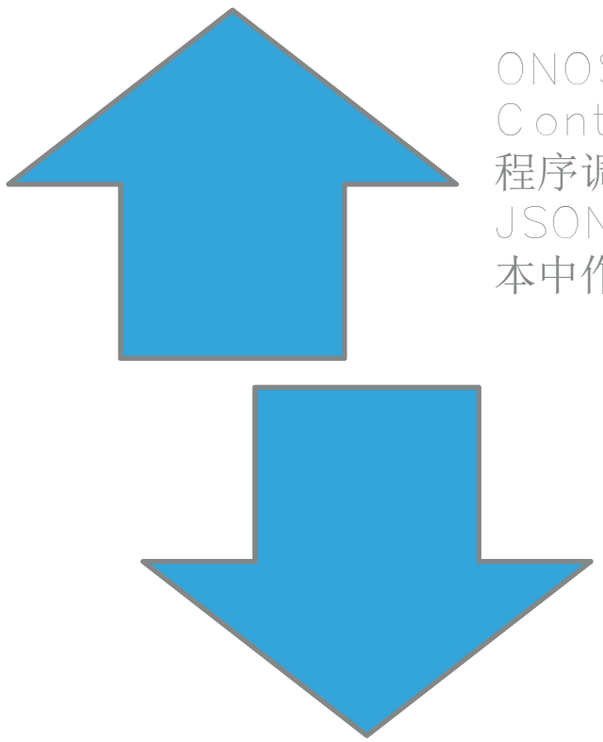
- 交换机ID+入端口ID+出端口ID——确定交换机端到端之间的不同路径。而传统的基于IP的路由追踪监测方式只能监控3层路径，无法区分Port-channel里的多条链路。
- 链路利用率——交换机端到端的不同路径的链路利用率既可以用于基础的监控，也可以用于选择新流量的路径选择，而不是盲目地将流量导向任意等价路径上。
- 延迟——交换机端到端的不同路径的链路延迟既可以用于基础的监控，也可以用于时间敏感类业务的智能路由。


针对数据包内容定义header
header
数据包
实例化
header实例
parser
根据hedaer定义字段确定偏移量，并提取数据包中的数据
将提取数据存储在实例中
存储了数据的header实例从流水线ingress进入
ingress
pipeline
egress
Table规定进行匹配的字段及匹配方式
table 1
table 2
...
table 3
flow control
Flow control中规定match-action表的应用顺序

FLOWLET和ECMP/WCMP



- 传统的路由协议都是采用单路径路由的方式
- 到达一个目的地有多条相同度量值的路由项(路由路径)
- 将所有链路权重视为相等，在链路上等比例传递流量
- 按照预设权重，按照不同比例在链路上传递流量



ONOS1.6版新增了“BMv2 Device Context Service”北向接口，应用程序调用此接口指定bmv2运行时的JSON配置。Bmv2接口将在下个版本中作为ONOS的核心接口。

ONOS南向通过thrift与bmv2通信，ONOS1.6版中修改了P4项目中的simple_switch模块，添加了对连接SDN控制器的支持。

内容

- onos-bmv2和p4c-bmv2编译脚本
- 构建基于onos-bmv2仿真网络环境的mininet脚本
- P4示例程序
- 基于mininet的bmv2调试工具

网络交换芯片技术

- ASIC

专用集成电路，面向特定需求。体积小，性能强，成本低，功能固化，不支持数据平面编程。代表：SDN白盒交换机。

- CPU

通用集成电路，功能灵活，性能受CPU限制，效率低，支持数据平面编程。代表：OpenvSwitch。

- NP

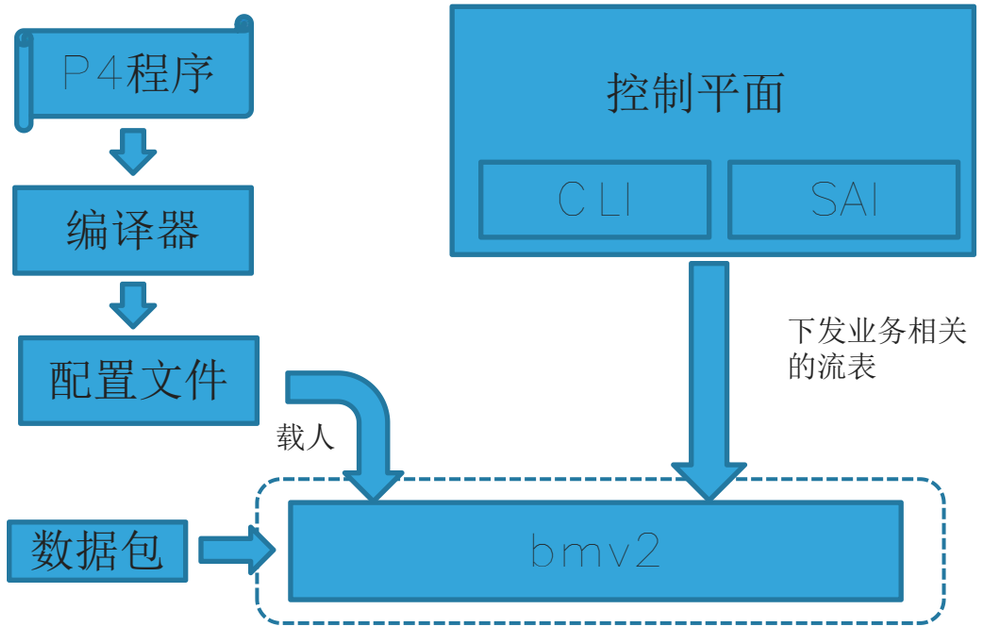
网络处理器，功能灵活，性能较高，成本高，功耗高，支持数据平面编程。代表：华为S12700系列。

- FPGA

现场可编程门阵列功能灵活，数据转发性能低，支持数据平面编程。代表：NetFPGA。

- PISA

协议无关交换机架构，功能灵活，性能强，支持数据平面编程。代表：Tofino。



P4定义的数据平面

Tips

- 目的交换机将数据导出到远程服务器上的网管系统或大数据分析工具
- 目的交换机将数据反馈到源交换机，以作为源交换机处理其他数据参考

下发流表方式

- Bmv2的管理CLI
- 调用SAI接口