

# **Programmable Dataplane using P4 Language**

Ramanand Shankarling

Rakesh Musalay

Naga Ganesh Kurapati

# Introduction

- ▶ P4 - Programming Protocol-Independent Packet Processors
- ▶ 3 major aspects
  - **Target independence**
  - **Protocol independence**
  - **Reconfigurability**

# Key components

## ► Headers

- describe sequence and structure of a series of fields.

## ► Parsers

- specify how to identify headers and valid header sequences within packets.

## ► Tables

- perform packet processing using fields for match.

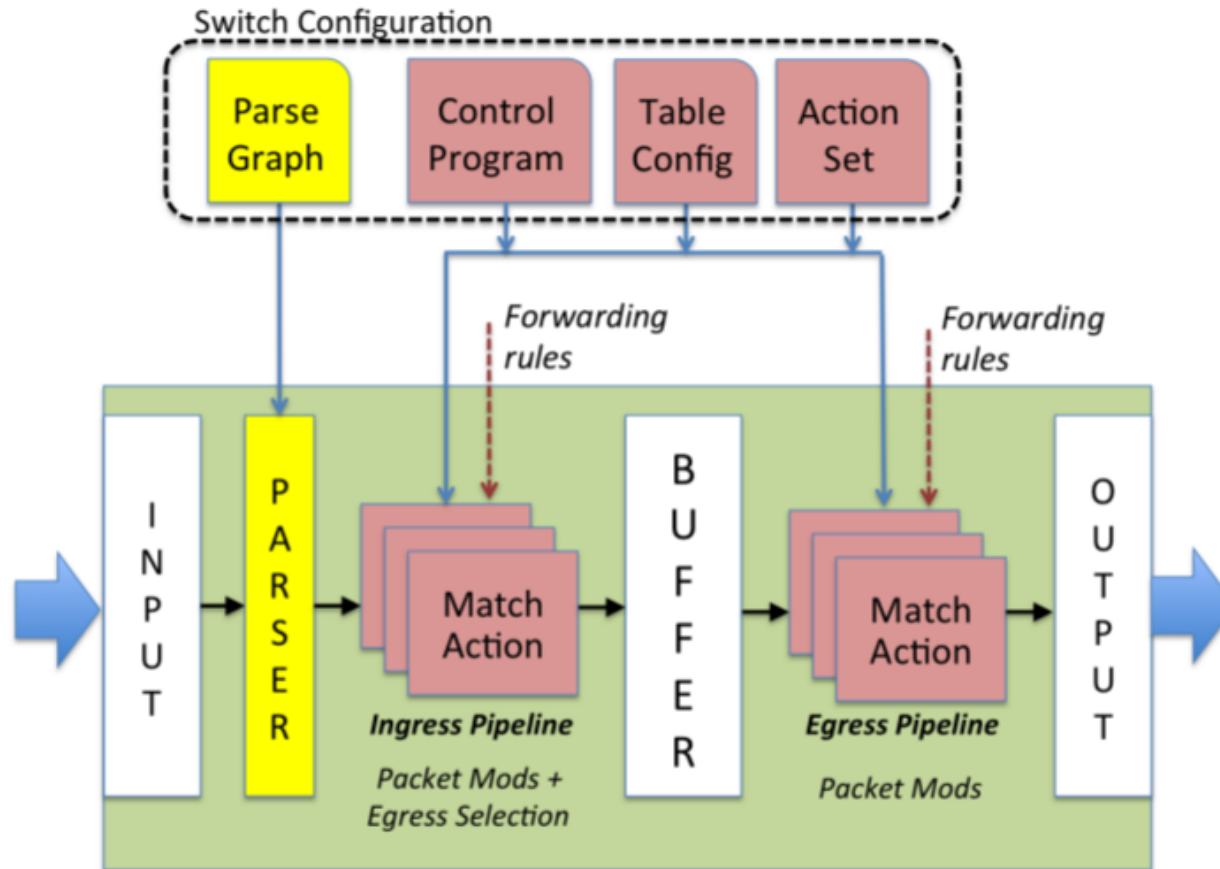
## ► Actions

- support for construction of complex actions from simpler protocol-independent primitives.

## ► Control programs

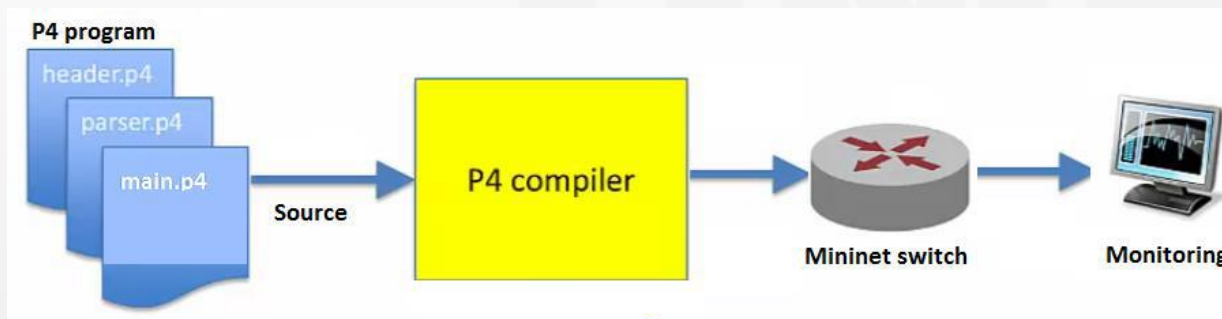
- determine the order of match + action tables that are applied to a packet.

# Abstract forwarding model



# Project Idea

- ▶ Design packet encapsulate/de-capsulate modules that can process packet headers and perform the following tasks applications and controllers
  - Identify video streams from/to a streaming server
  - Identify YouTube streams
  - Identify OpenSSL flows
  - Record and report statistics regarding the above flows



# Tools and programs

## Mist streaming server

- Open-source multimedia streaming server
- Supported formats and protocols : HLS, HDS, Smooth Streaming, RTMP, RTSP, MP3, FLV
- Works on multiple platforms and provides web interface for configuration management

# Mist server web interface

MistServer MI - Mozilla Firefox

MistServer MI

localhost:4242/#ganeshkurapati&http://localhost:4242/api@Preview&ex1

MistServer

Overview  
Protocols  
Streams  
**Preview**  
Push  
Triggers  
Logs  
Statistics  
Server Stats  
Disconnect  
Guides  
Tools

You're watching MP4 (html5/video/mp4) through HTML5 video player.

Use player: Automatic

Use source: Automatic

**Player log:**

```
[23:14:00] Retrieving stream info from http://localhost:8080/info_ex1.js
[23:14:00] Stream info was loaded successfully.
[23:14:00] Checking available players.
[23:14:00] Checking HTML5 video player (priority: 1).
[23:14:00] This browser does not support html5/application/vnd.apple.mpegurl
via http://localhost:8080/hls/ex1/index.m3u8
[23:14:00] Found a workina combo: HTML5 video player with html5/video/mp4 @
```

**Meta information**

Type: Pre-recorded (VoD)

Tracks:

	Track 2	Track 1
Audio	Codec: AAC	Codec: H264
	Duration: 00:04:47.463	Duration: 00:04:47.100
	00:00:00 to 00:04:47.463	00:00:00 to 00:04:47.100
	Peak bitrate: 8.3 KiB/s	Peak bitrate: 67.4 KiB/s
	Channels: 2	Size: 640 x 480 px
Video	Samplerate: 44 100 Hz	Framerate: 0 fps
	Language: unknown	Language: unknown

# TCP REPLAY

## ► TCP REWRITE

- Provides packet editing functionality for pcap files

- Rewriting Source & Destination MAC addresses

```
tcprewrite --enet-dmac=mac1 --enet-smac=mac2 --  
infile=input.pcap --outfile=output.pcap
```

- Modify Source and destination IP Addresses

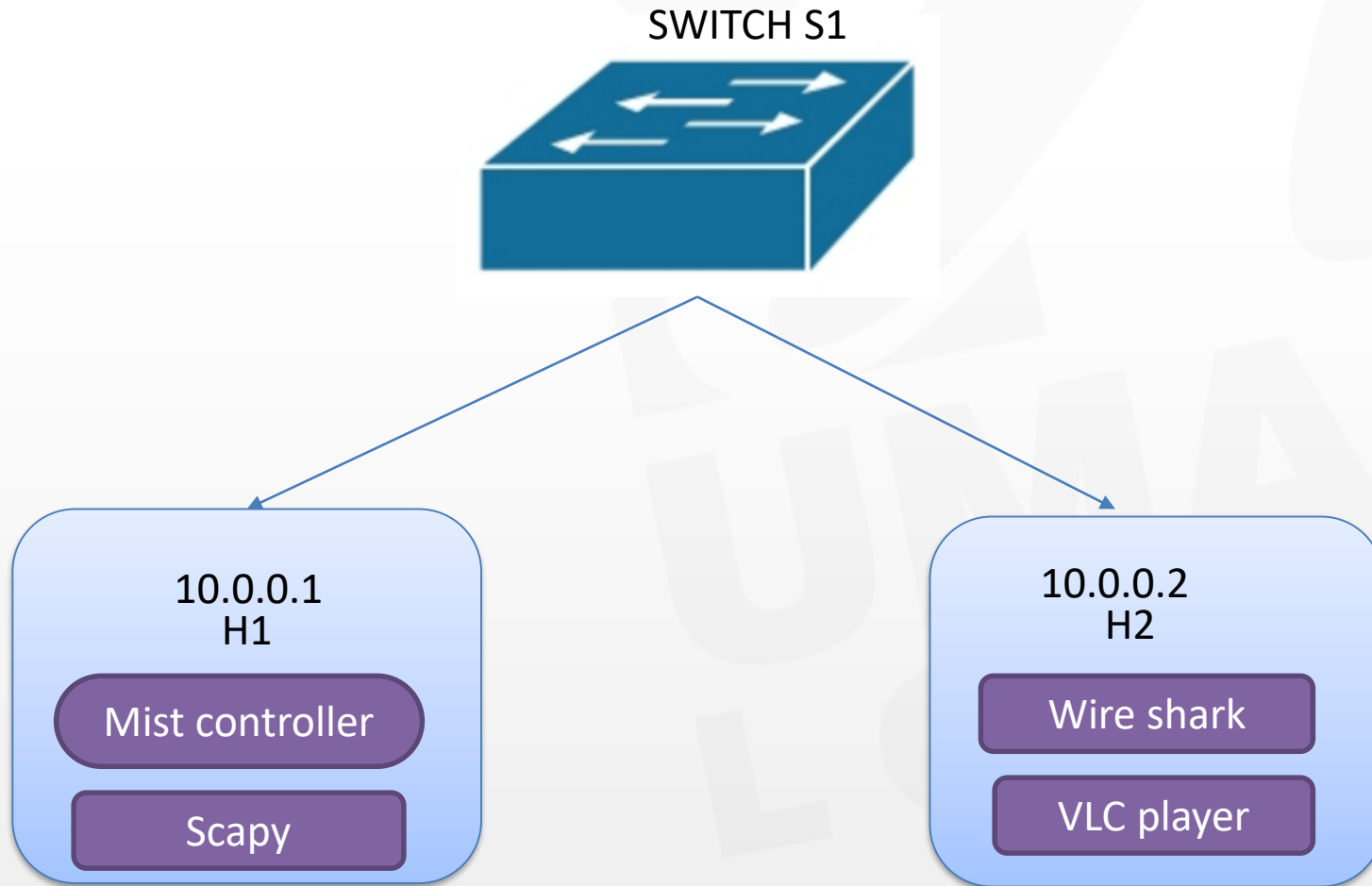
```
tcprewrite --endpoints=ipv4-1:ipv4-2 --cachefile=example.cache --  
infile=example.pcap --outfile=new.pcap
```



# SCAPY

- ▶ Forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies
- ▶ Replay modified packets and send them to the switch s1
- ▶ Packets are sent using the command  
`sendp(rdpicap("/tmp/pcapfile"))`

# Environment setup



# DEMO

UMASS  
LOWELL

# P4 Program

## headers.p4 and parser.p4

```
header_type tcp_t {  
    fields {  
        srcPort : 16;  
        dstPort : 16;  
        seqNo : 32;  
        ackNo : 32;  
        dataOffset : 4;  
        res : 4;  
        flags : 8;  
        window : 16;  
        checksum : 16;  
        urgentPtr : 16;  
    }  
}
```

```
header_type udp_t {  
    fields {  
        srcPort : 16;  
        dstPort : 16;  
        length : 16;  
        checksum : 16;  
    }  
}
```

```
#define IP_PROTOCOLS_TCP 6  
#define IP_PROTOCOLS_UDP 17  
  
parser parse_ipv4 {  
    extract(ipv4);  
    return select(latest.protocol) {  
        IP_PROTOCOLS_TCP : parse_tcp;  
        IP_PROTOCOLS_UDP : parse_udp;  
        default: ingress;  
    }  
}  
  
header tcp_t tcp;  
  
parser parse_tcp {  
    extract(tcp);  
    return ingress;  
}  
  
header udp_t udp;  
  
parser parse_udp {  
    extract(udp);  
    return ingress;  
}
```

# P4 Program

## Counters

```
action on_miss() {  
}  
  
counter c_quic {  
    type: packets;  
    instance_count: 32;  
}  
  
action count_quic(){  
    count(c_quic, 1);  
}  
  
counter c_ssl {  
    type: packets;  
    instance_count: 32;  
}  
  
action count_ssl(){  
    count(c_ssl, 1);  
}
```

```
counter c_rtmp {  
    type: packets;  
    instance_count: 32;  
}  
  
action count_rtmp(){  
    count(c_rtmp, 1);  
}
```

# P4 Program

## Tables

```
table quic_count_src {
  reads {
    udp.srcPort : exact;
  }
  actions {
    on_miss;
    count_quic;
    _drop;
  }
  size : 256;
}
```

```
table quic_count_dst {
  reads {
    udp.dstPort : exact;
  }
  actions {
    on_miss;
    count_quic;
    _drop;
  }
  size : 256;
}
```

```
table ssl_rtmp_count_src {
  reads {
    tcp.srcPort : exact;
  }
  actions {
    on_miss;
    count_ssl;
    count_rtmp;
    _drop;
  }
  size: 256;
}
```

```
table ssl_rtmp_count_dst {
  reads {
    tcp.dstPort : exact;
  }
  actions {
    on_miss;
    count_rtmp;
    _drop;
  }
  size: 256;
}
```

# P4 Program

## ingress and egress

```
control ingress {
    apply(ipv4_lpm);
    apply(forward);
    if(valid(udp)){
        apply(quic_count_src){
            on_miss{
                apply(quic_count_dst);
            }
        }
    }//if
    if(valid(tcp)){
        apply(ssl_rtmp_count_src){
            on_miss{
                apply(ssl_rtmp_count_dst);
            }
        }
    }//if
}

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

# Add or Delete Table Entries

## Bash files

```
python ../../cli/pd_cli.py -p simple_router -i p4_pd_rpc.simple_router -s $PWD/tests/pd_thrift:  
$PWD/../../../../testutils -m "add_entry quic_count_src 443 count_quic" -c localhost:22222  
python ../../cli/pd_cli.py -p simple_router -i p4_pd_rpc.simple_router -s $PWD/tests/pd_thrift:  
$PWD/../../../../testutils -m "add_entry quic_count_dst 443 count_quic" -c localhost:22222|
```

```
python ../../cli/pd_cli.py -p simple_router -i p4_pd_rpc.simple_router -s $PWD/tests/pd_thrift:  
$PWD/../../../../testutils -m "delete_entry quic_count_src 0" -c localhost:22222  
python ../../cli/pd_cli.py -p simple_router -i p4_pd_rpc.simple_router -s $PWD/tests/pd_thrift:  
$PWD/../../../../testutils -m "delete_entry quic_count_dst 0" -c localhost:22222|
```

```
python ../../cli/pd_cli.py -p simple_router -i p4_pd_rpc.simple_router -s $PWD/tests/pd_thrift:  
$PWD/../../../../testutils -m "add_entry quic_count_src 443 _drop" -c localhost:22222  
python ../../cli/pd_cli.py -p simple_router -i p4_pd_rpc.simple_router -s $PWD/tests/pd_thrift:  
$PWD/../../../../testutils -m "add_entry quic_count_dst 443 _drop" -c localhost:22222
```



# Conclusion

- ▶ Successful implementation of program using P4 language to parse and identify specific type of packets, depending on which further actions can be defined.
- ▶ Future work
  - The counter data to be fetched for traffic analysis and implement reporting.
  - Contextual routing.

# References

- ▶ [https://en.wikipedia.org/wiki/P4\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/P4_(programming_language))
- ▶ P4: Programming Protocol-Independent Packet Processors - P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker; CCR July 2014
- ▶ <http://p4.org/>