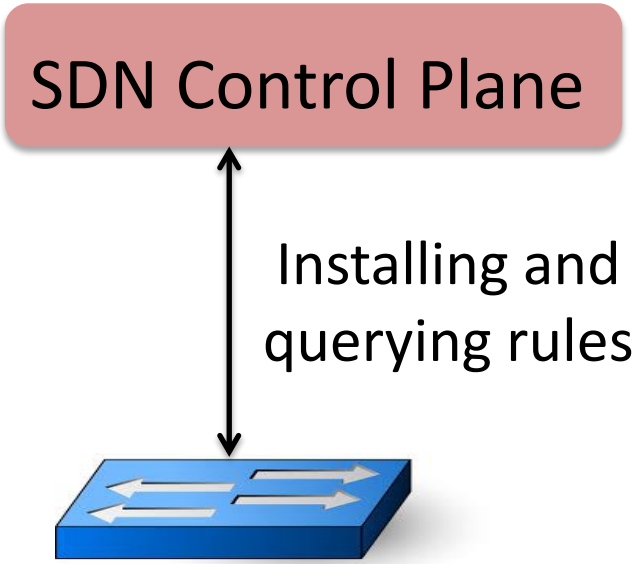
P4 for Media

(Programming Protocol-Independent Packet Processors)

Thomas Edwards
VP Engineering & Development
FOX Networks Engineering & Operations

OpenFlow-based SDN



SDN gives operators programmatic control over their networks.
The control plane is physically separate from the forwarding plane.
One control plane can control multiple forwarding devices.
OpenFlow is one SDN API.

Match/Action Rule Examples

Match: in_port=1
Action: output port 2

Match: ip,nw_dst=10.0.0.3
Action: output port 3

Match: nw_dst=10.*.*.*,udp_dst=2000
Action: drop

FOX NETWORKS ENGINEERING & OPERATIONS

Proliferation of OF header fields

Version	Date	# Headers
OF 1.0	Dec 2009	12 (Ethernet, TCP/IPv4)
OF 1.1	Feb 2011	15 (+MPLS)
OF 1.2	Dec 2011	36 (+ARP, ICMP, IPv6)
OF 1.3	Jun 2012	40
OF 1.4	Oct 2013	41

But still not enough (e.g., VXLAN, NVGRE, STT, …)

Future Switches…

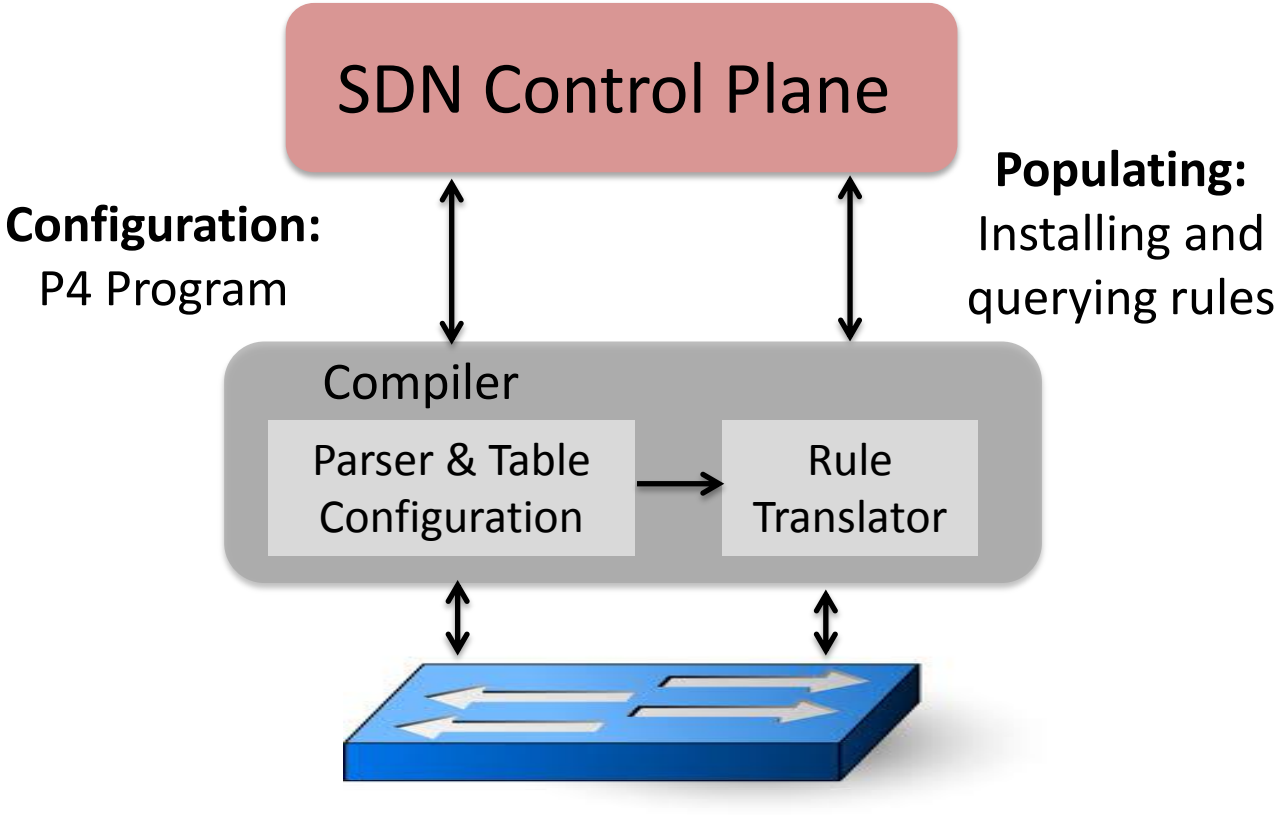
- Configurable packet parser
  - Not tied to a specific header format
- Flexible match+action tables
  - Multiple tables (in series and/or parallel)
  - Able to match on any field
- General packet-processing primitives
  - Copy, add, remove, and modify
  - For both header fields and meta-data

How about a Language? P4

- Protocol independent
- Configure a packet parser
  - Define a set of typed match+action tables
- Target independence
  - Program at high level without knowledge of switch details
  - Rely on compiler to configure the target switch
- Reconfigurability
  - Change parsing and processing in the field
- http://p4.org

FOX NETWORKS ENGINEERING & OPERATIONS

P4 Model


SDN Control Plane
Configuration:
P4 Program
Populating:
Installing and
querying rules
Compiler
Parser & Table
Configuration
Rule
Translator

Header Definitions

```
header_type ethernet_t {
        fields {
            dstAddr : 48;
            srcAddr : 48;
            etherType : 16;
        }
    }
```

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

```
header_type rtp_t {
        fields {
            version : 2;
            padding : 1;
            extension : 1;
            CSRC_count : 4;
            marker : 1;
            payload_type : 7;
            sequence_number : 16;
            timestamp : 32;
            SSRC : 32;
        }
    }
```

Parser

```
parser ethernet {
    extract(ethernet);
    return select(ethernet.etherType) {
        0x800:   parse_ipv4;
        0x8100: parse_vlan;
        0x88F7: parse_ptp;
        0xF000: parse_my_protocol;
        default: ingess;
      }
   }
}
```

FOX NETWORKS ENGINEERING & OPERATIONS

Typed Tables & Action Functions

```
table schedule_table {
    reads {
        ipv4.dstAddr: exact;
        rtp.timestamp: range;
    }
    actions {
        take_video;
        _drop;
     }
     size : 16384;
}
```

Ipv4.dstAddr	rtp.timestamp	action	Dst_ip
239.1.1.1	0x0000->0x0002	take_video	239.3.3.3
239.2.2.2	0x0003->0x0004	take_video	239.3.3.3
239.1.1.1	0x0005->0x0006	take_video	239.3.3.3
default	default	default	_drop

```
action take_video(dst_ip) {
    modify_field(standard_metadata.egress_spec,1);
    modify_field(ipv4.dstAddr,dst_ip);
}
```

FOX NETWORKS ENGINEERING & OPERATIONS

Results…

Looking at output of switch port 1…

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	239.3.3.3	RTP	62	PT=ITU-T G.711 PCMU, SSRC=0x0, Time=1
2	1.769727000	192.168.1.1	239.3.3.3	RTP	62	PT=ITU-T G.711 PCMU, SSRC=0x0, Time=2
3	3.333013000	192.168.2.2	239.3.3.3	RTP	62	PT=ITU-T G.711 PCMU, SSRC=0x0, Time=3
4	4.912446000	192.168.2.2	239.3.3.3	RTP	62	PT=ITU-T G.711 PCMU, SSRC=0x0, Time=4
5	6.434700000	192.168.1.1	239.3.3.3	RTP	62	PT=ITU-T G.711 PCMU, SSRC=0x0, Time=5
6	8.698775000	192.168.1.1	239.3.3.3	RTP	62	PT=ITU-T G.711 PCMU, SSRC=0x0, Time=6

Other P4 apps:

- ECMP based on RTP sequence #
- Multicast replication + per output port NAT

https://github.com/FOXNEOAdvancedTechnology

FOX NETWORKS ENGINEERING & OPERATIONS