P4: specifying data planes

netdev0.1

Ottawa, February 15, 2015

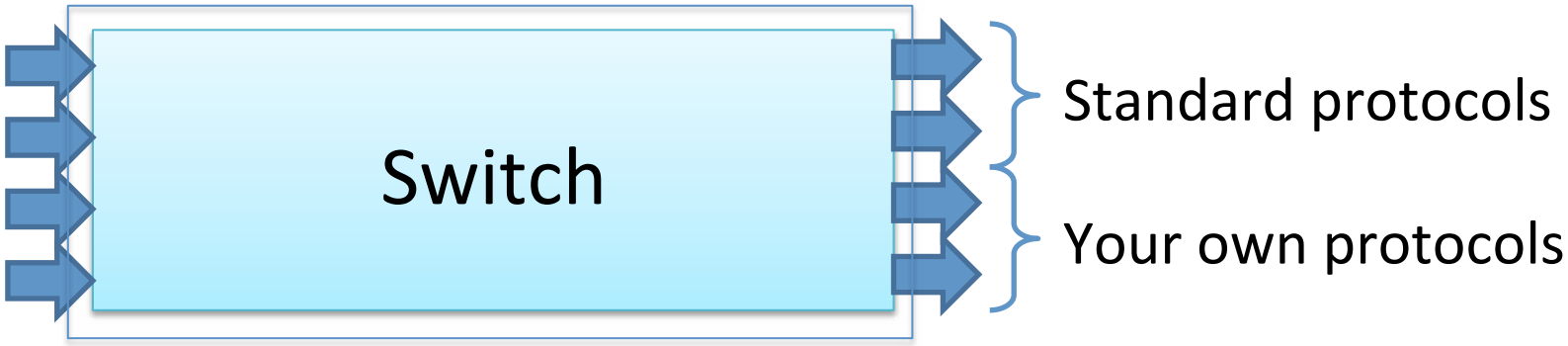
Mihai Budiu

Proceedings of netdev 0.1, Feb 14-17, 2015, Ottawa, On, Canada

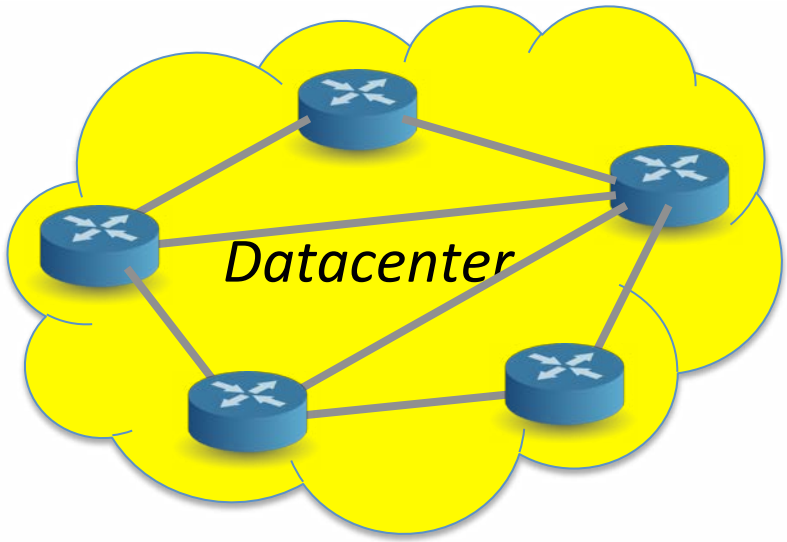
P4 Language Consortium

- "Open for participation by any individual or corporation"
- No membership fees.
- Language spec v1.0.1 published
- Coming soon:
  - 3/2015 - FOSS release of a reference P4 implementation
- http://p4.org

What do we want to achieve?



Currently most useful if you have your own network playground



Benefits

- Implement protocols quickly
  - VxLAN: 175 lines of code
  - NVGRE: 183 lines of code
- Low overhead (high speed)
- Flexible forwarding policies
- Improved monitoring, and troubleshooting
- Change functionality with software upgrades
- Use only what you need

P4 Scope

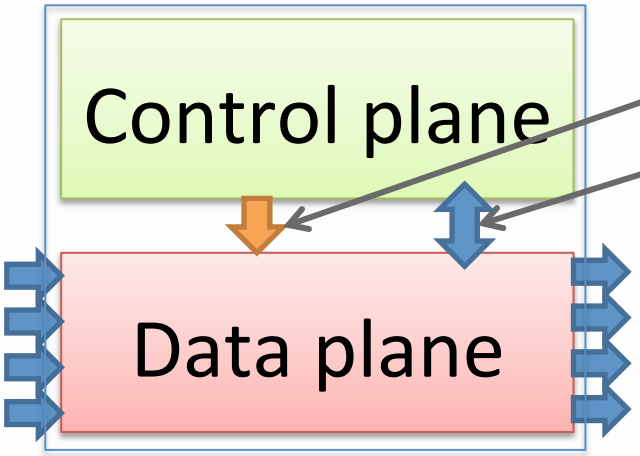
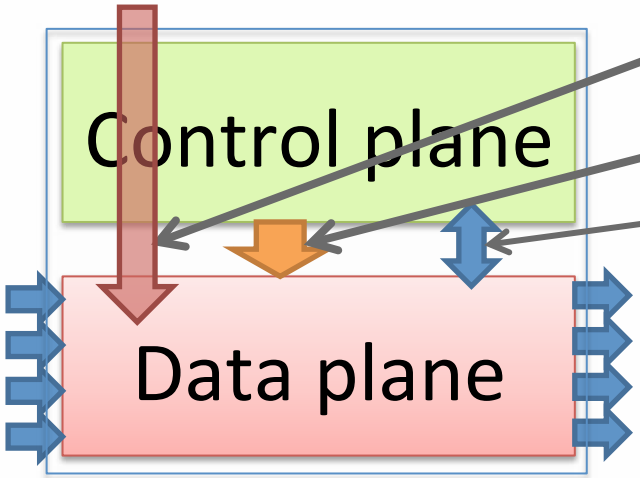
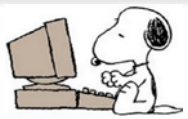


Table mgmt.

Control traffic (Pkts.)

P4-defined switch



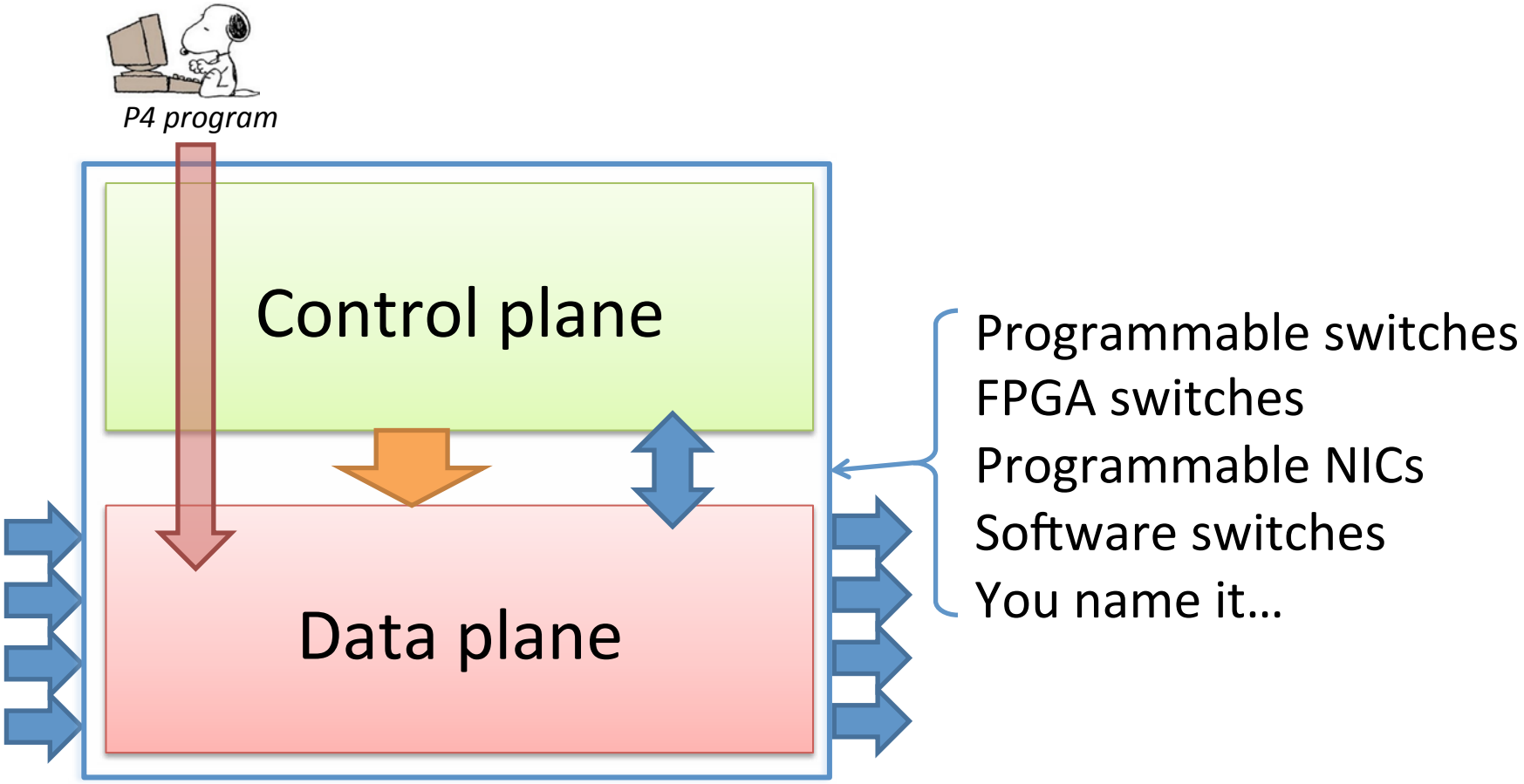


P4 program

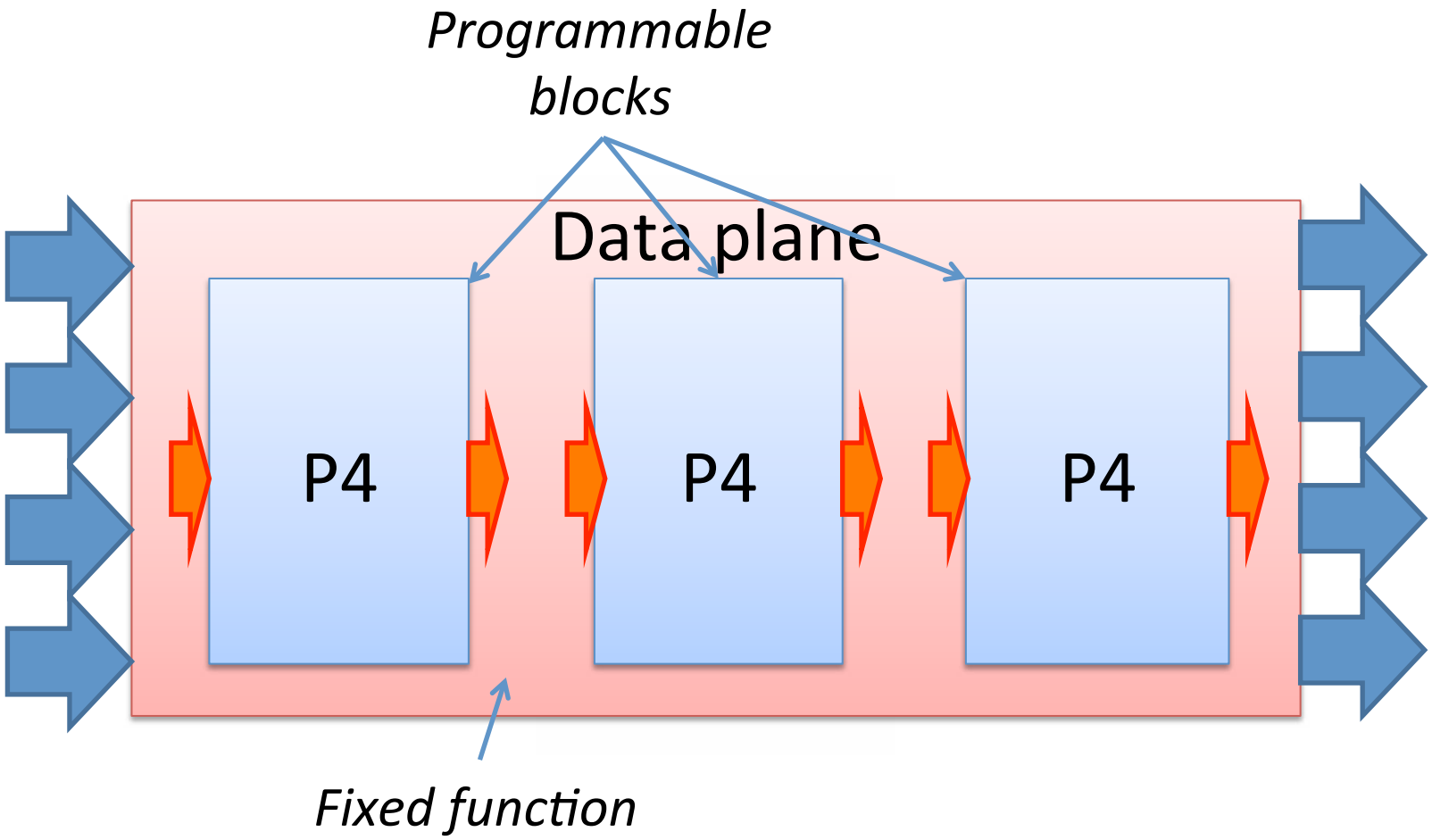
P4 table mgmt.

Control traffic (Pkts.)

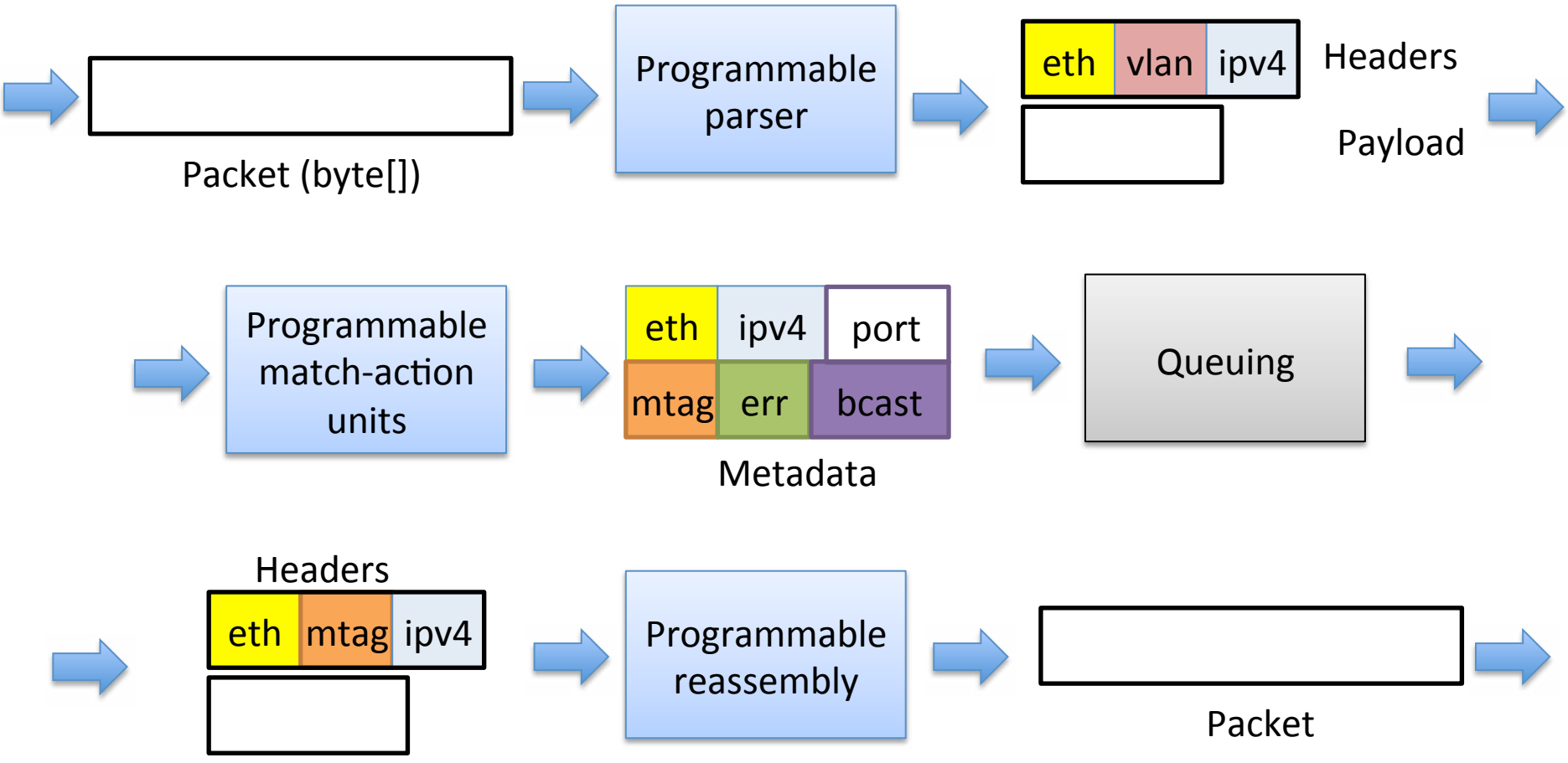
Q: Which data plane?
A: Any data plane!



Data plane programmability



How does it work?



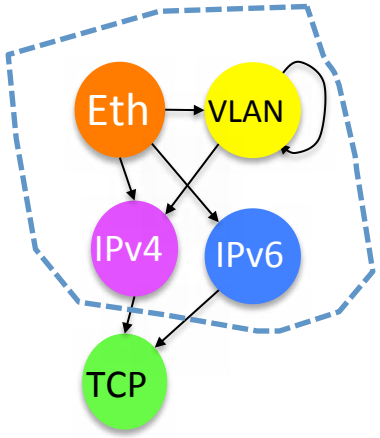
P4 language

Programmable parser	State-machine; bitfield extraction
Programmable match-action units	Table lookup and update; bitfield manipulation; control flow
Programmable reassembly	Bitfield assembly

No: memory (pointers), loops, recursion, floating point

Parsing = State machines

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




```
parser parse_ethernet {
  extract(ethernet);
  return select(latest.etherType) {
    0x8100 : parse_vlan;
    0x800  : parse_ipv4;
    0x86DD : parse_ipv6;
  }
}
```

Match

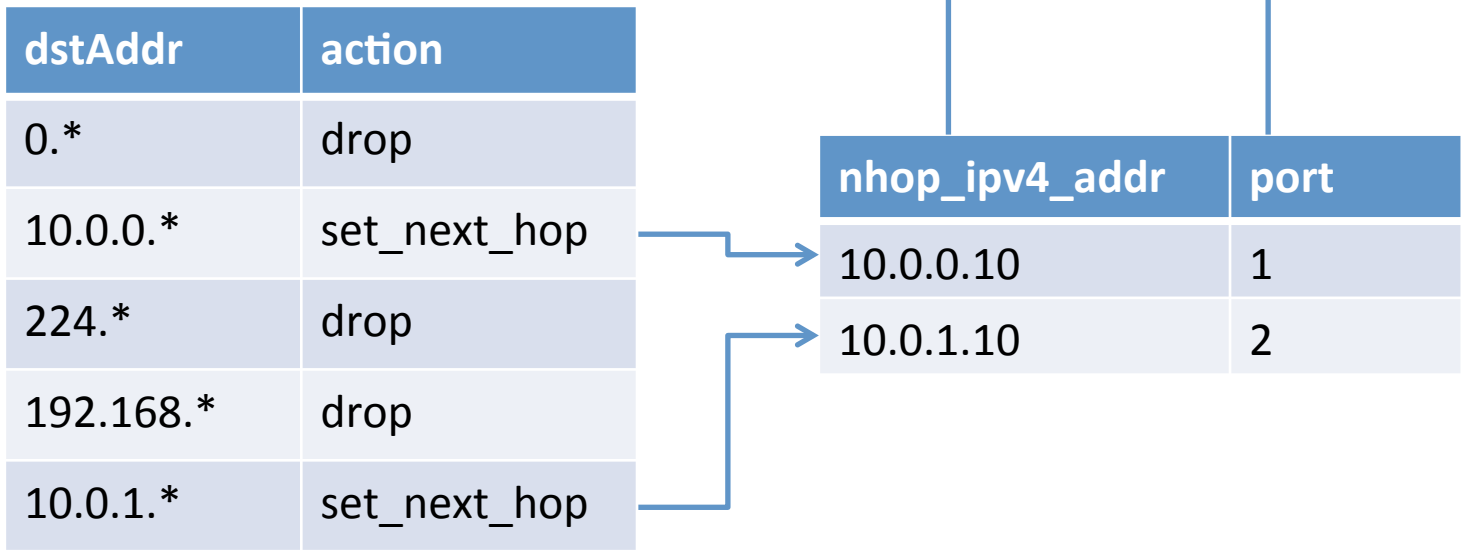
```
table ipv4_lpm
{
    reads {
        ipv4.dstAddr : lpm;
    }
    actions {
        set_next_hop;
        drop;
    }
}
```

Lookup key

dstAddr	action
0.*	drop
10.0.0.*	set_next_hop
224.*	drop
192.168.*	drop
10.0.1.*	set_next_hop

Actions

```
action set_nhop(nhop_ipv4_addr, port)
{
    modify_field(metadata.nhop_ipv4_addr, nhop_ipv4_addr);
    modify_field(standard_metadata.egress_port, port);
    add_to_field(ipv4.ttl, -1);
}
```

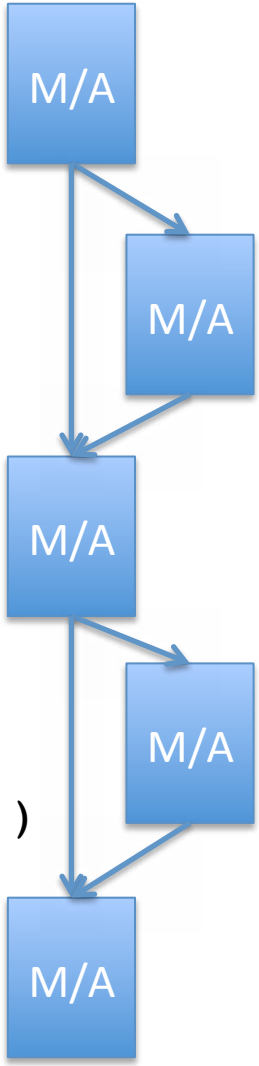


dstAddr	action
0.*	drop
10.0.0.*	set_next_hop
224.*	drop
192.168.*	drop
10.0.1.*	set_next_hop

nhop_ipv4_addr	port
10.0.0.10	1
10.0.1.10	2

Control-Flow

```
control ingress
{
    apply(port);
    if (valid(vlan_tag[0])) {
        apply(port_vlan);
    }
    apply (bridge_domain);
    if (valid(mpls_bos)) {
        apply(mpls_label);
    }
    retrieve_tunnel_vni();
    if (valid(vxlan) or valid(genv) or valid(nvgre))
    {
        apply(dest_vtep);
        apply(src_vtep);
    }
}
```



Reassembly

- Driven by header types

- `add_header(ipv6);`
- `remove_header(vlan);`

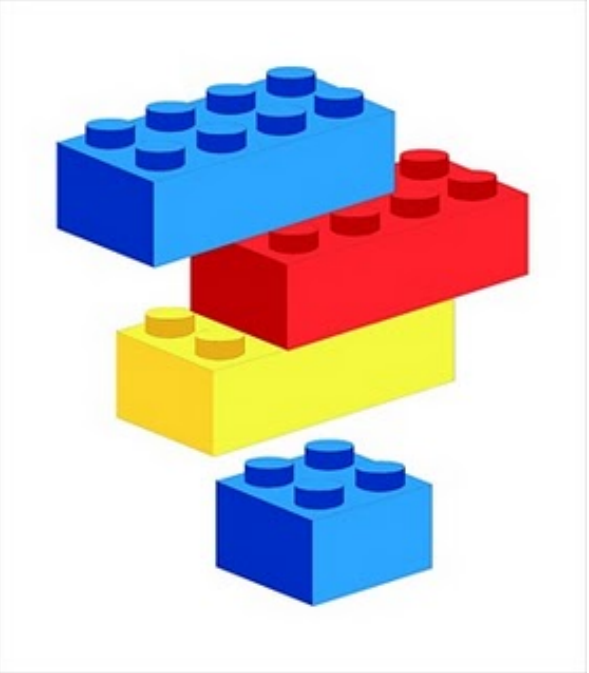
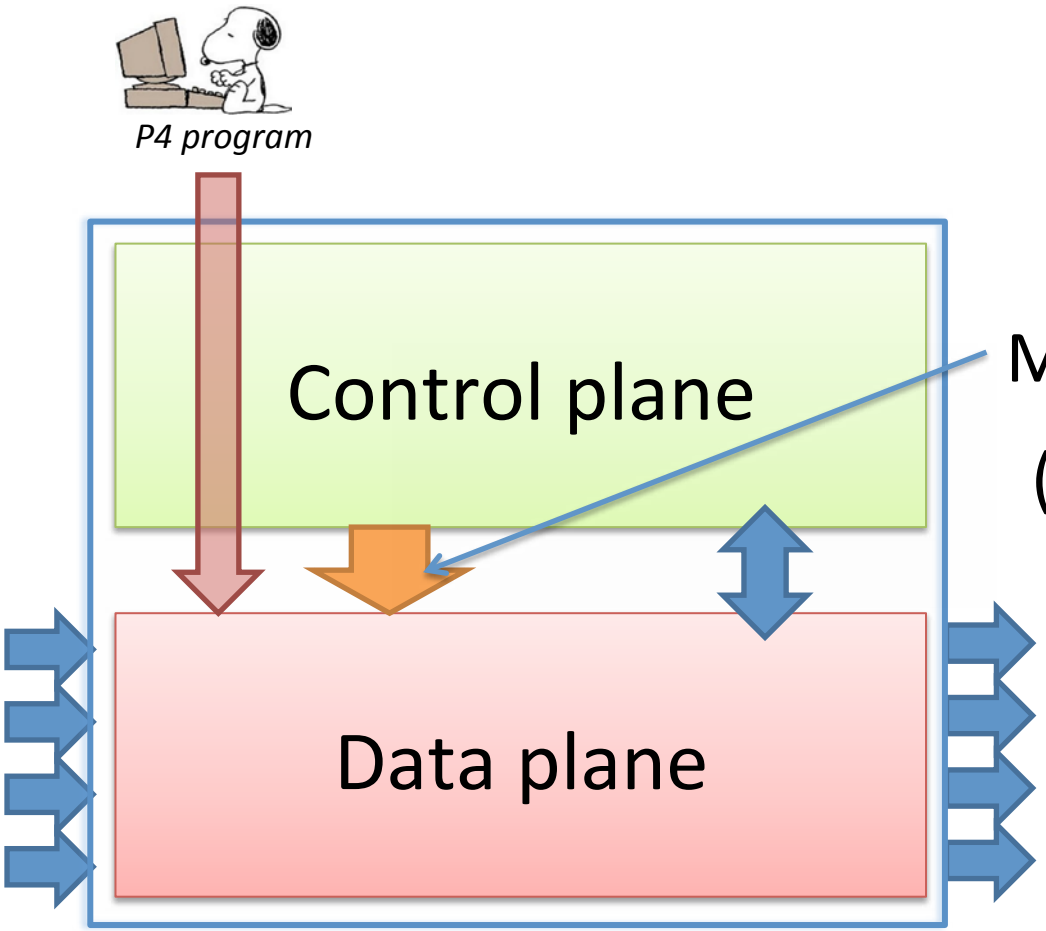


Table contents management



Manage tables contents

(Tied to P4 program)

P4 Summary

- Simple language
  - Parsing, bit-field manipulation, table lookup, control flow, packet reassembly
- Efficient execution (high speed switching)
- Simple cost model
- Abstract resources
- Portable
- Expressive:
  - New protocols, forwarding policies, monitoring and instrumentation

Creating a Programming Language Interface in a place where there wasn't one.