Hypothesis

A high level description of a dataplane can be implemented in BPF

P4 Overview

P4 Program Sections

program.p4

Data Declarations

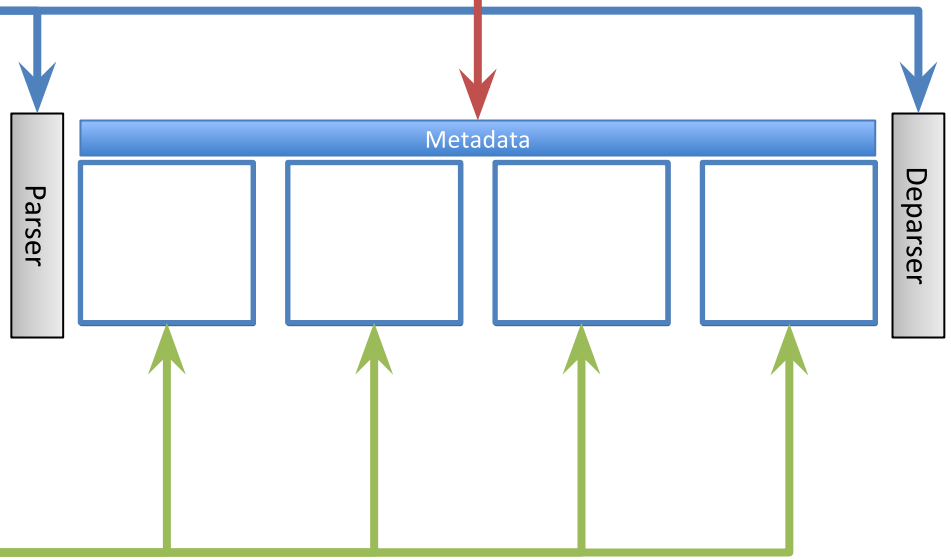
```
header_type ethernet_t   { ... }
header_type l2_metadata_t { ... }

header   ethernet_t   ethernet;
header   vlan_tag_t   vlan_tag[2];
metadata l2_metadata_t l2_meta;
```

Parser Program

```
parser parse_ethernet {
  extract(ethernet);
  return switch(ethernet.ethertype) {
    0x8100 : parse_vlan_tag;
    0x0800 : parse_ipv4;
    0x8847 : parse_mpls;
    default: ingress;
}
```

Control Flow Program

```
table port_table { ... }

control ingress {
  apply(port_table);
  if (l2_meta.vlan_tags == 0) {
    process_assign_vlan();
  }
}
```



P4 Syntax Example: Metadata

Example: Declaring Metadata

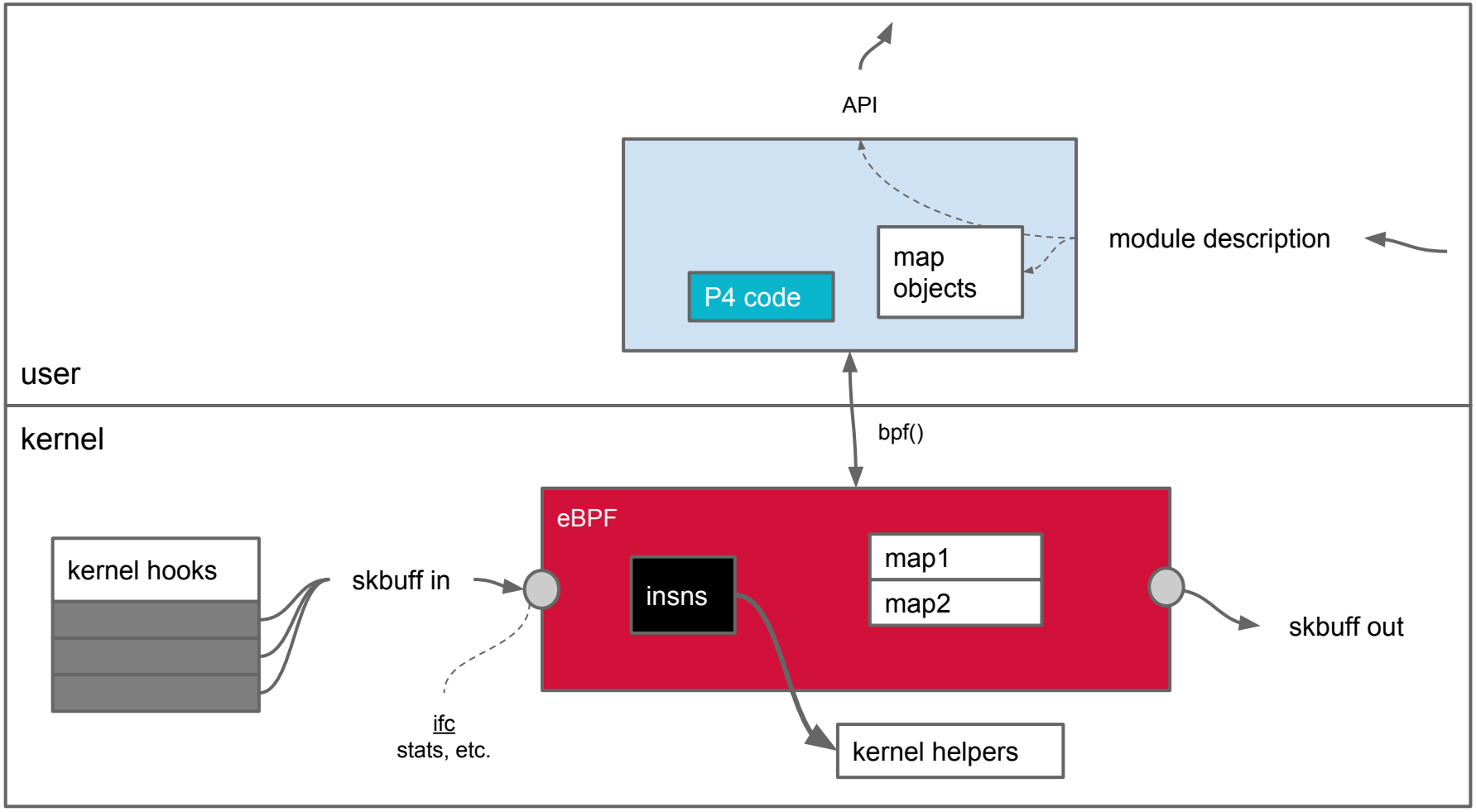
```
header_type ingress_metadata_t {
    fields {
     /* Inputs */
        ingress_port           : 9;  /* Available prior to parsing    */
        packet_length          : 16; /* Might not be always available */
        instance_type          : 2;  /* Normal, clone, recirculated   */
        ingress_global_tstamp  : 48;
        parser_status          : 8;  /* Parsing Error */

     /* Outputs from Ingress Pipeline */
        egress_spec            : 16;
        queue_id               : 9;
   }
}

metadata ingress_metadata_t ingress_metadata;
```

Metadata is a header too

Actual Metadata Instantiations

P4 to BPF

BPF Programs



Review

Instructions

- 10x 64bit registers
- 512B stack
- 1-8B load/store
- conditional jump
- arithmetic
- function call

Helper functions

- forward/clone/drop packet
- load/store packet data
- load/store packet metadata
- checksum (incremental)
- push/pop vlan
- access kernel mem (kprobes)

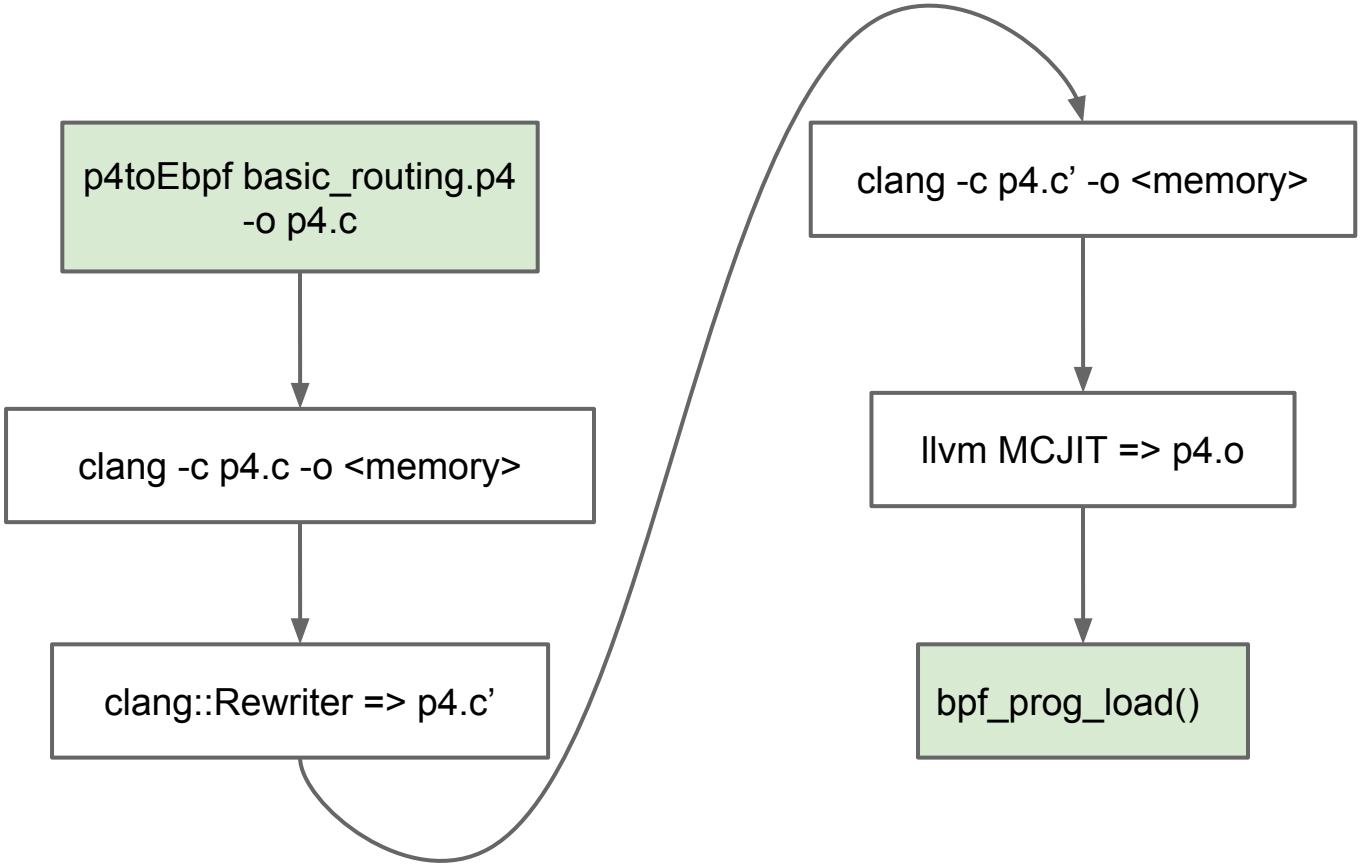
Data structures

- lookup/update/delete
  - in-kernel or from userspace
- hash, array, ...

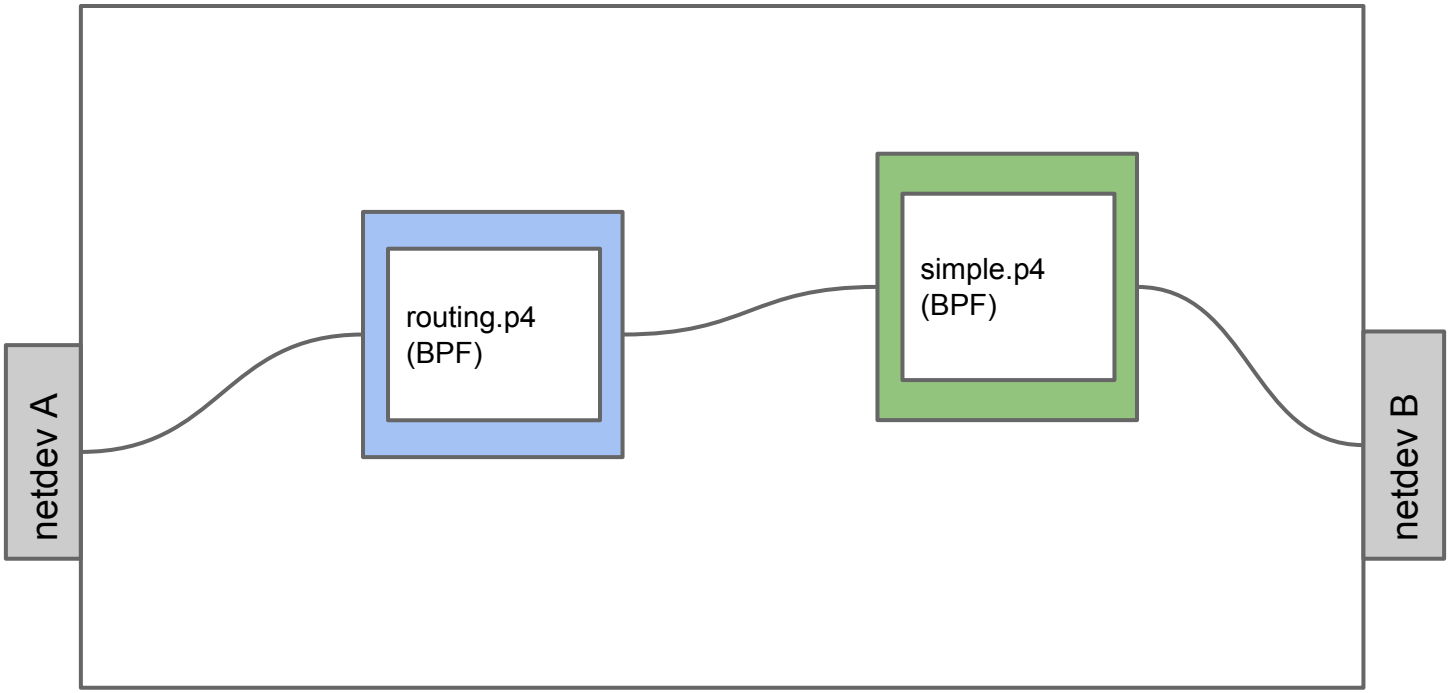
Review

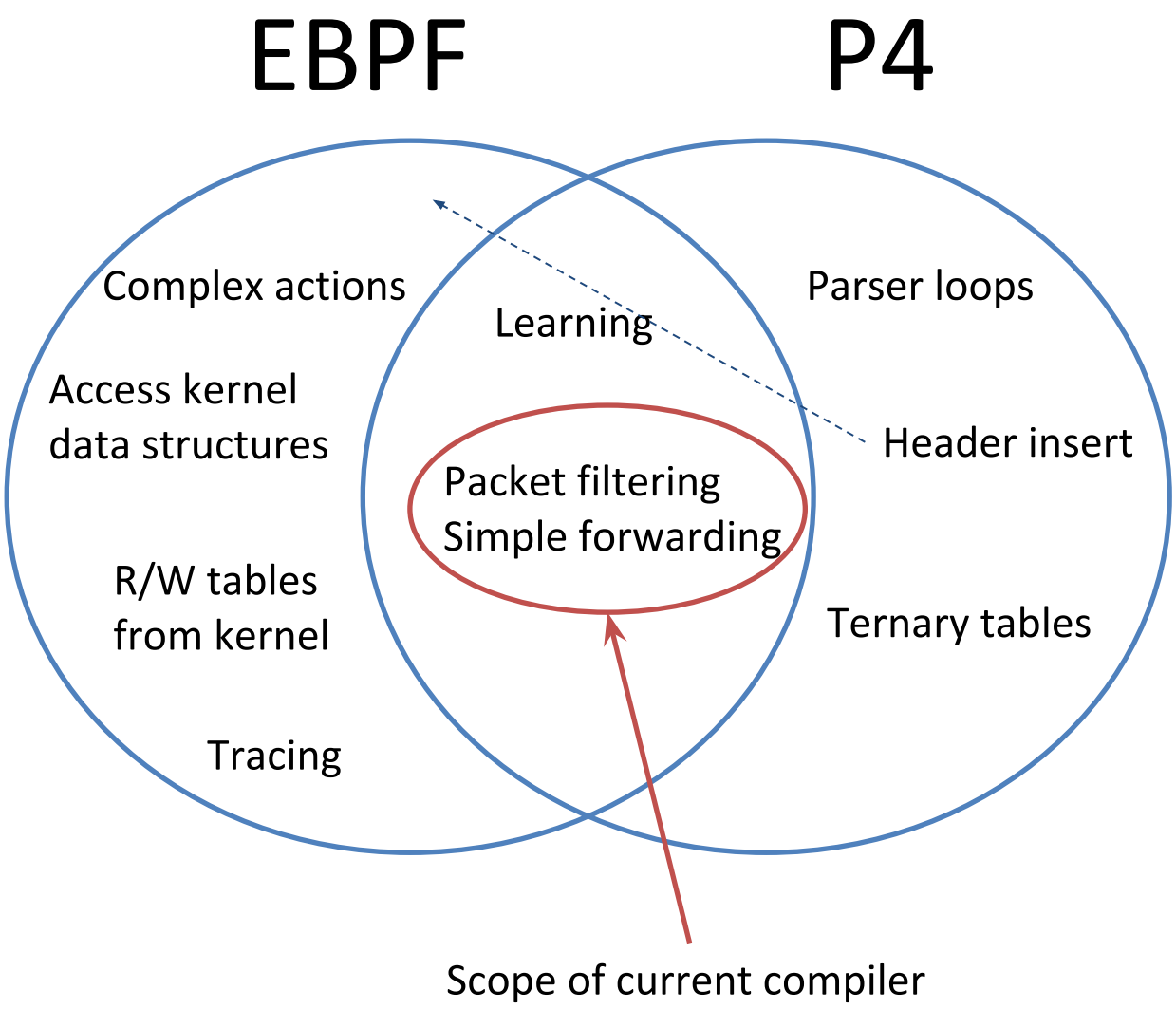
- Attach net programs at
  - SOL_SOCKET
    - PACKET_FANOUT_BPF
  - SO_REUSEPORT
  - tc ingress
  - tc egress
    - filter
    - action
    - clsact

P4 -> BPF Workflow



Composing Modules




EBPF
P4
Complex actions
Learning
Parser loops
Access kernel
data structures
Header insert
Packet filtering
Simple forwarding
R/W tables
from kernel
Ternary tables
Tracing
Scope of current compiler

TODO

- Checksum support
- Push/Pop header
- More table types (FIB integration)
- Frontend optimizations
- Many others...

References

https://github.com/iovisor/bcc/tree/master/src/cc/frontends/p4