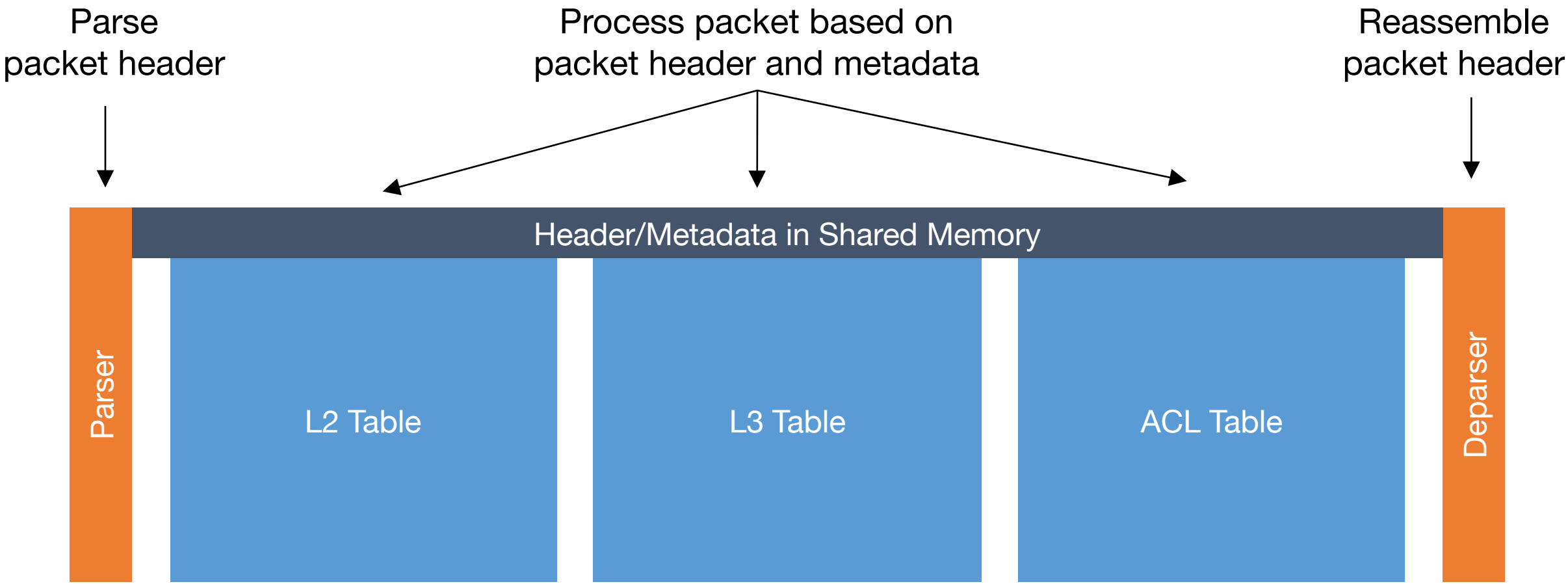
P4 Tutorial

Xin Jin

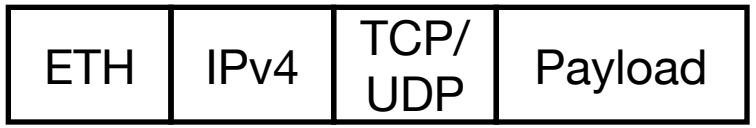
xinjin@cs.jhu.edu

Traditional Switch: Fixed Packet Processing

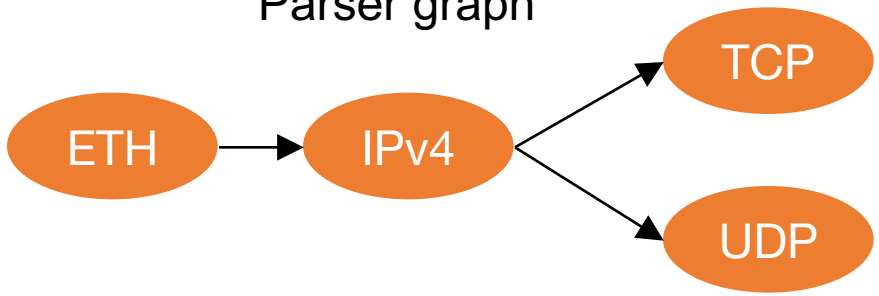


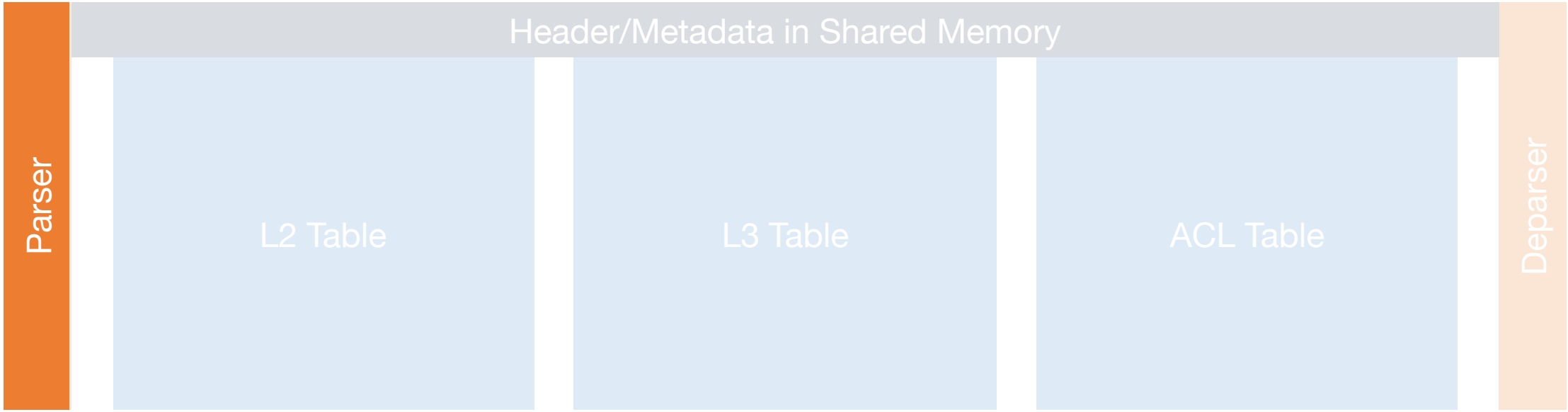
Traditional Switch: Fixed Packet Processing

Packet format



Parser graph





Traditional Switch: Fixed Packet Processing

Packet format

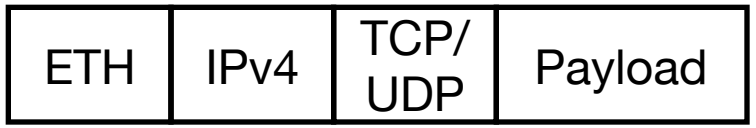
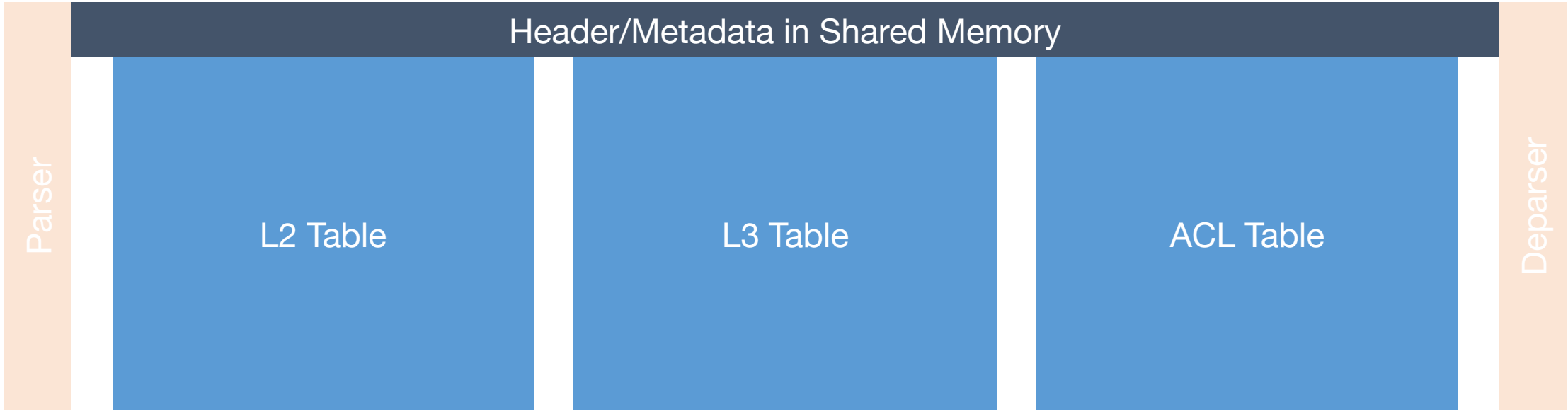


Table graph



Traditional Switch: Fixed Packet Processing

Packet format

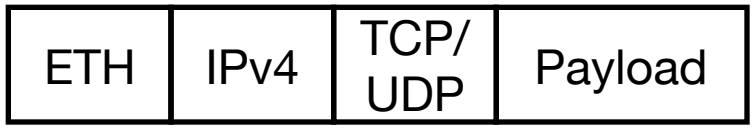
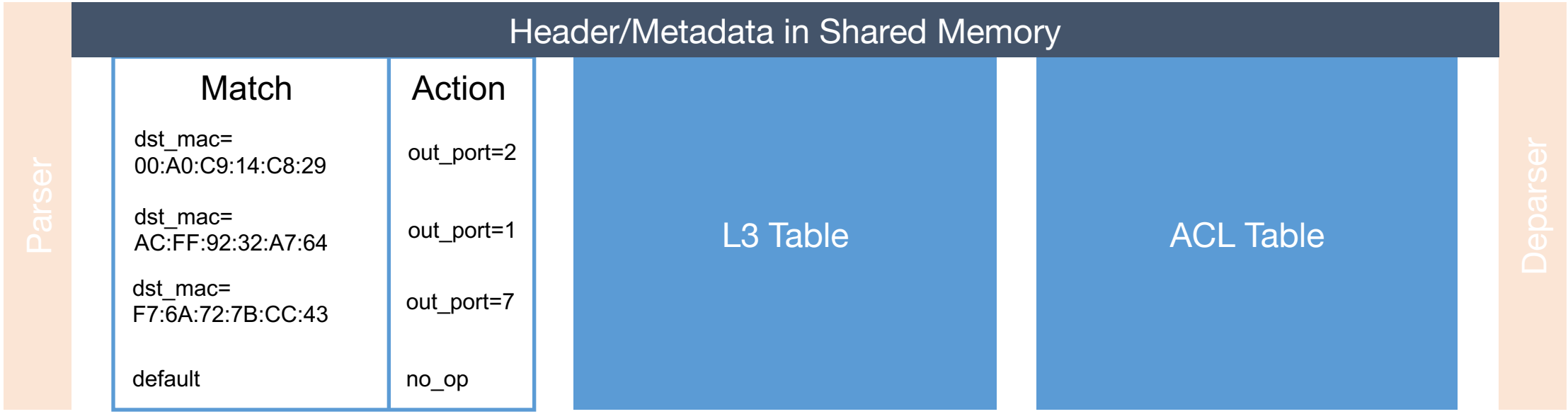


Table graph



Traditional Switch: Fixed Packet Processing

Packet format

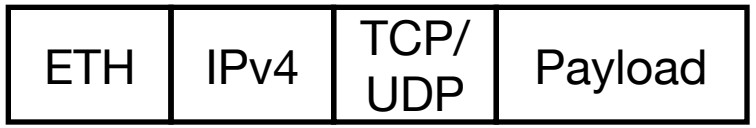
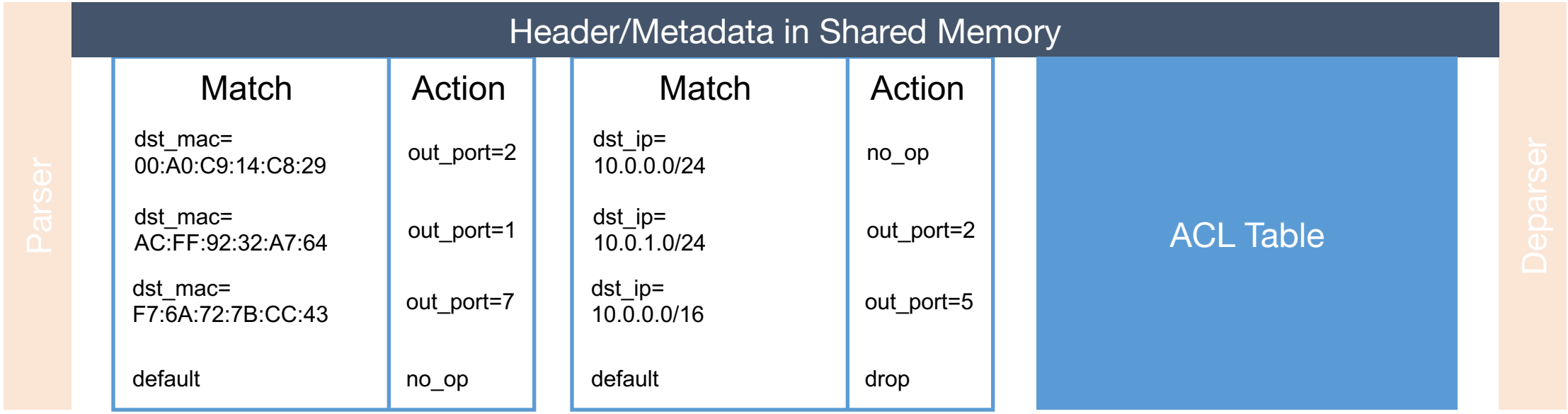


Table graph



Traditional Switch: Fixed Packet Processing

Packet format

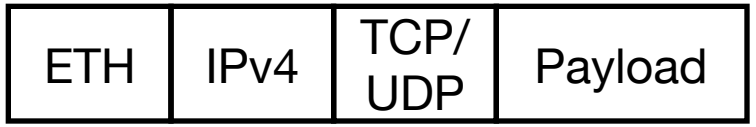


Table graph

Parser

Header/Metadata in Shared Memory

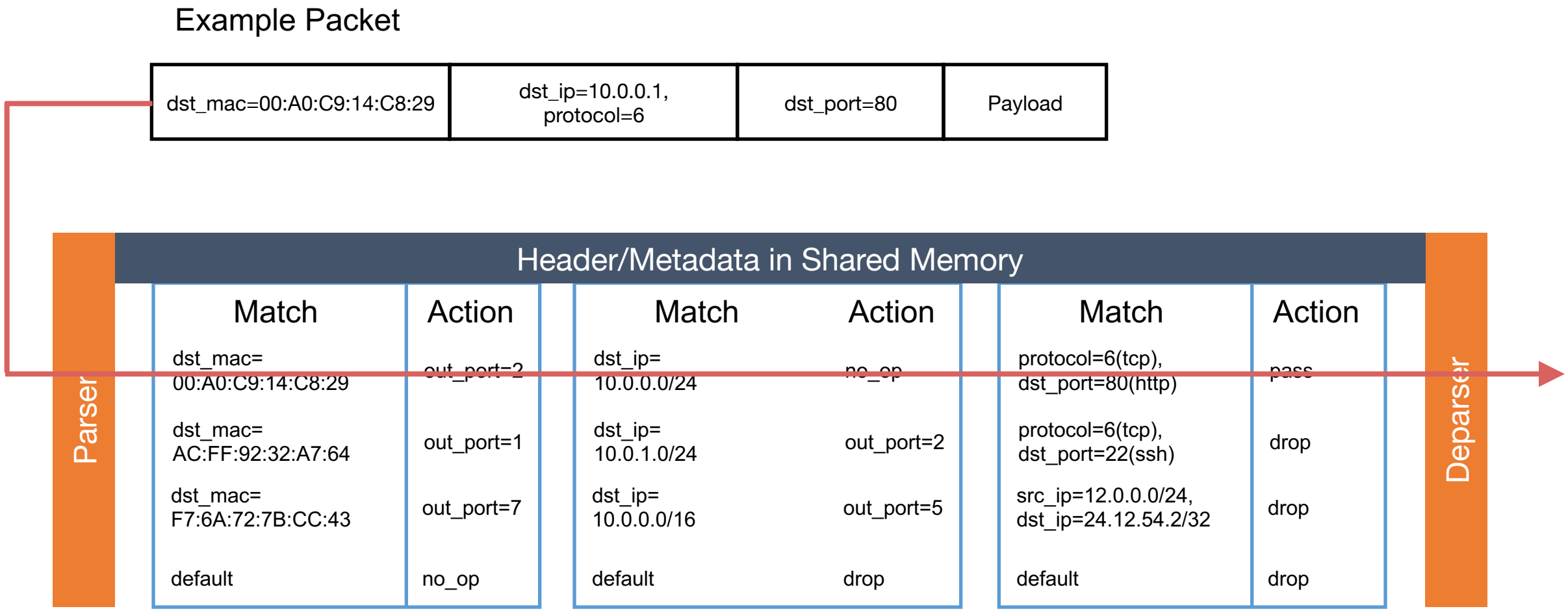
Match	Action
dst_mac= 00:A0:C9:14:C8:29	out_port=2
dst_mac= AC:FF:92:32:A7:64	out_port=1
dst_mac= F7:6A:72:7B:CC:43	out_port=7
default	no_op

Match	Action
dst_ip= 10.0.0.0/24	no_op
dst_ip= 10.0.1.0/24	out_port=2
dst_ip= 10.0.0.0/16	out_port=5
default	drop

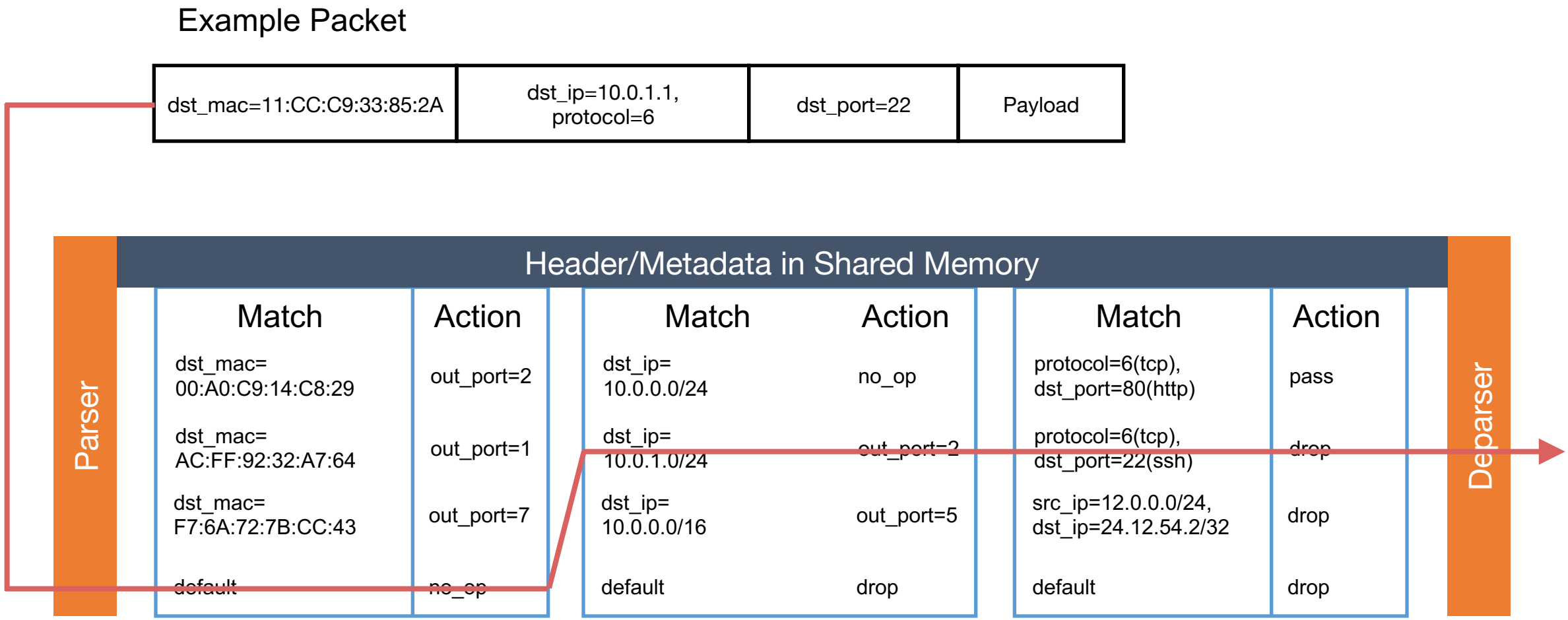
Match	Action
protocol=6(tcp), dst_port=80(http)	pass
protocol=6(tcp), dst_port=22(ssh)	drop
src_ip=12.0.0.0/24, dst_ip=24.12.54.2/32	drop
default	drop

Deparser

Traditional Switch: Fixed Packet Processing

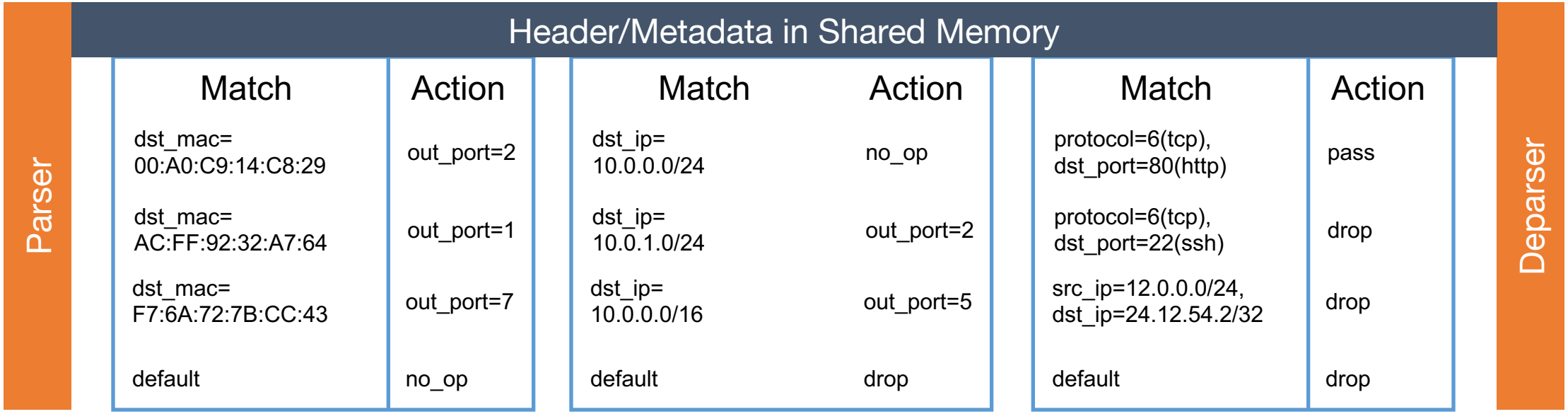


Traditional Switch: Fixed Packet Processing



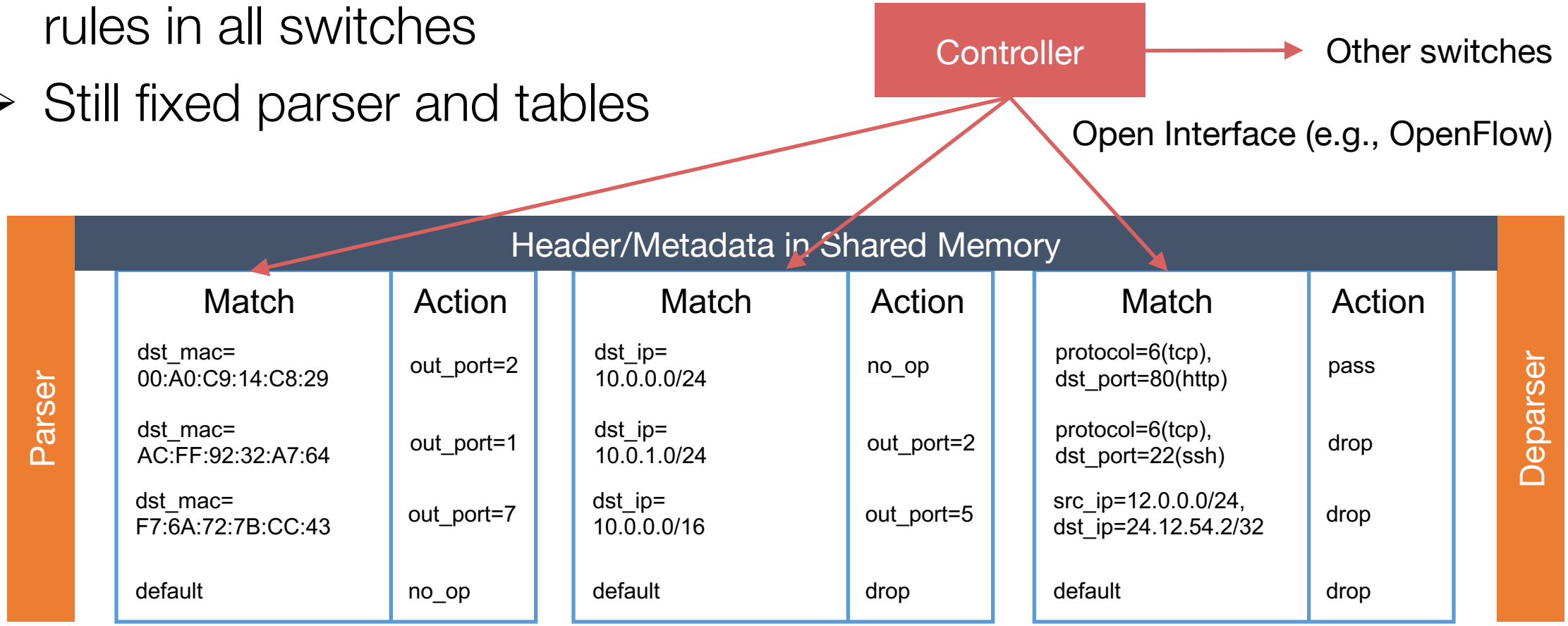
Traditional Switch: Fixed Packet Processing

- Fixed parser
- Fixed match-action tables
  - Rules in tables are generated by (proprietary) distributed network protocols or configured with (proprietary) switch management interface



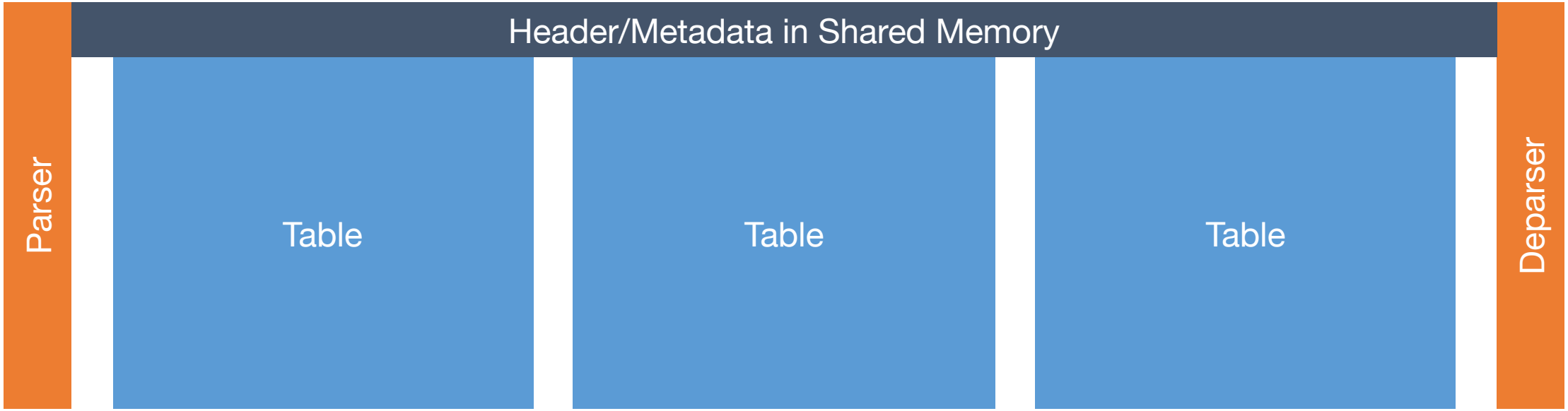
Software-Defined Networking: Centralized Control of Switches

- Centralized controller configures rules in all switches
- Still fixed parser and tables



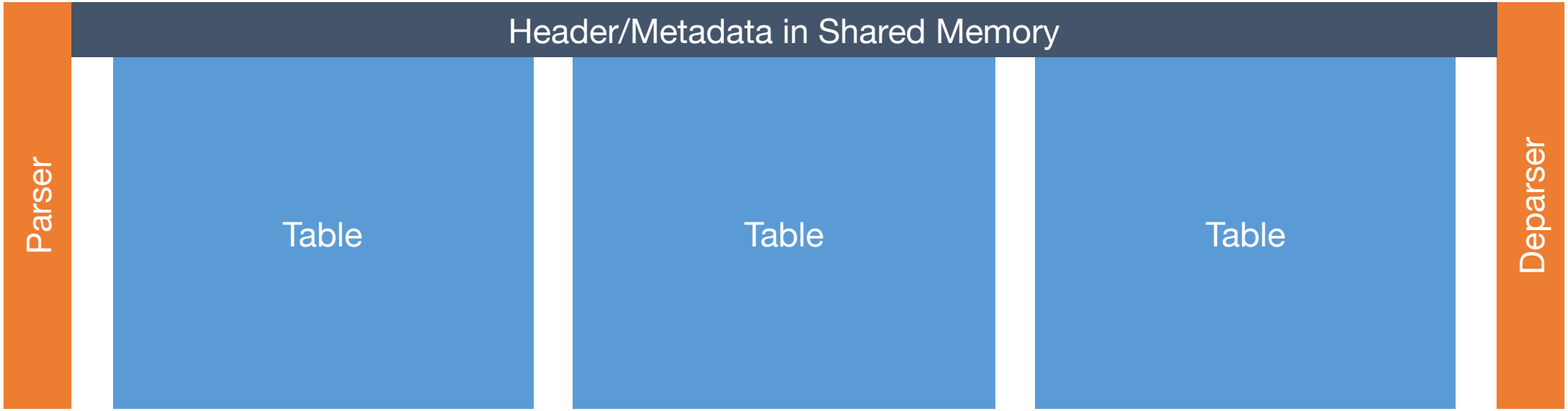
P4: Programmable Protocol-Independent Packet Processing

- Programmable parser: easily add new header
- Programmable table
  - Easily change match fields and actions
  - Easily change rules in the table



P4: Programmable Protocol-Independent Packet Processing

Add new features without changing the hardware



Example: Add IPv6 Support

Parser graph

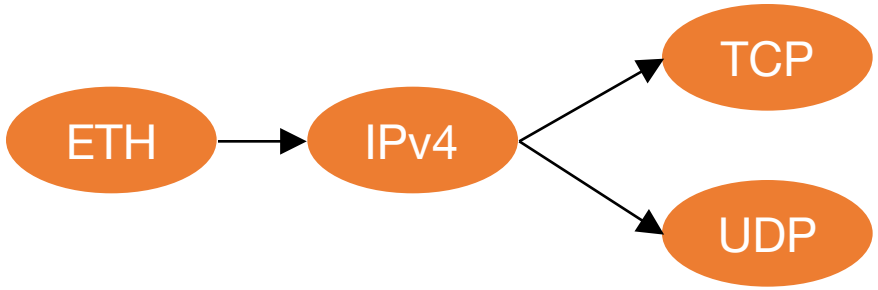
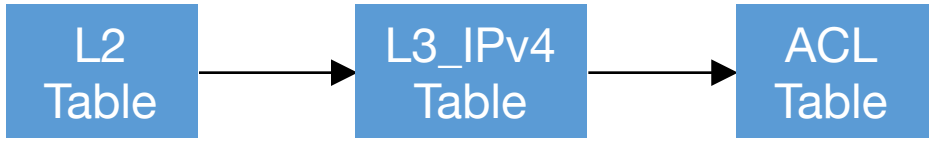
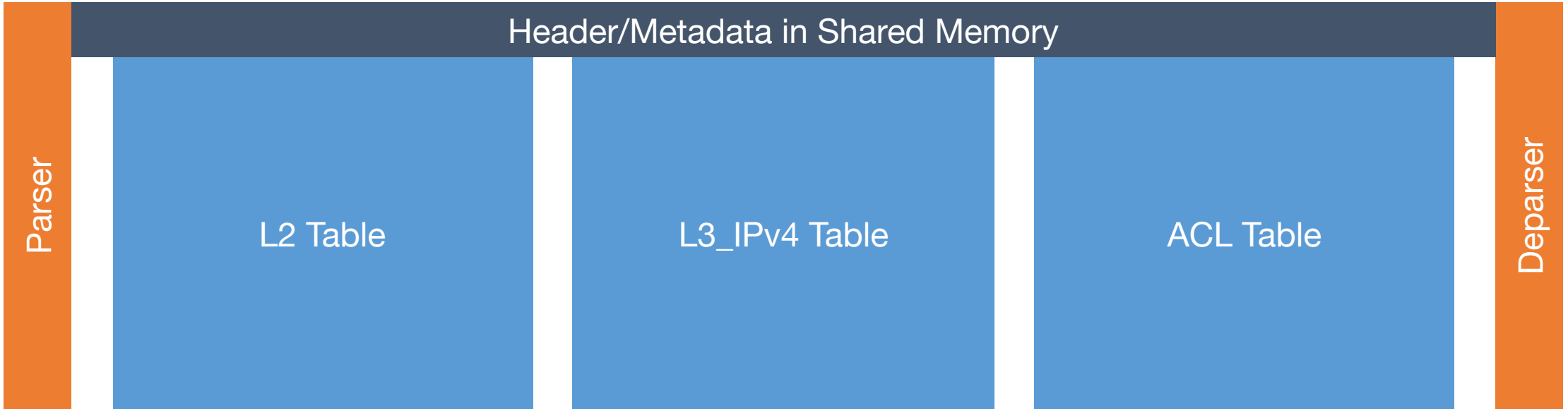


Table graph





Example: Add IPv6 Support

Parser graph

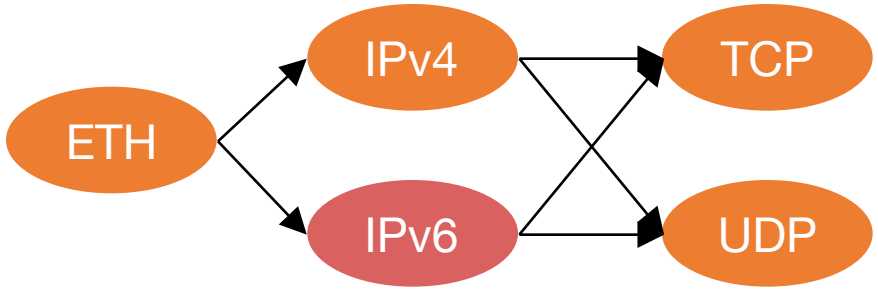
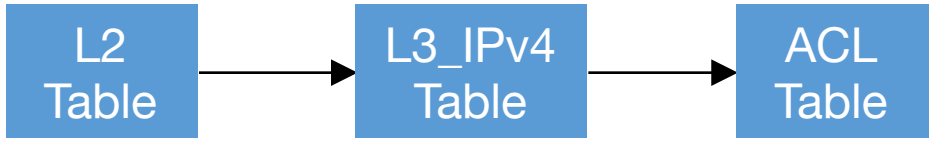
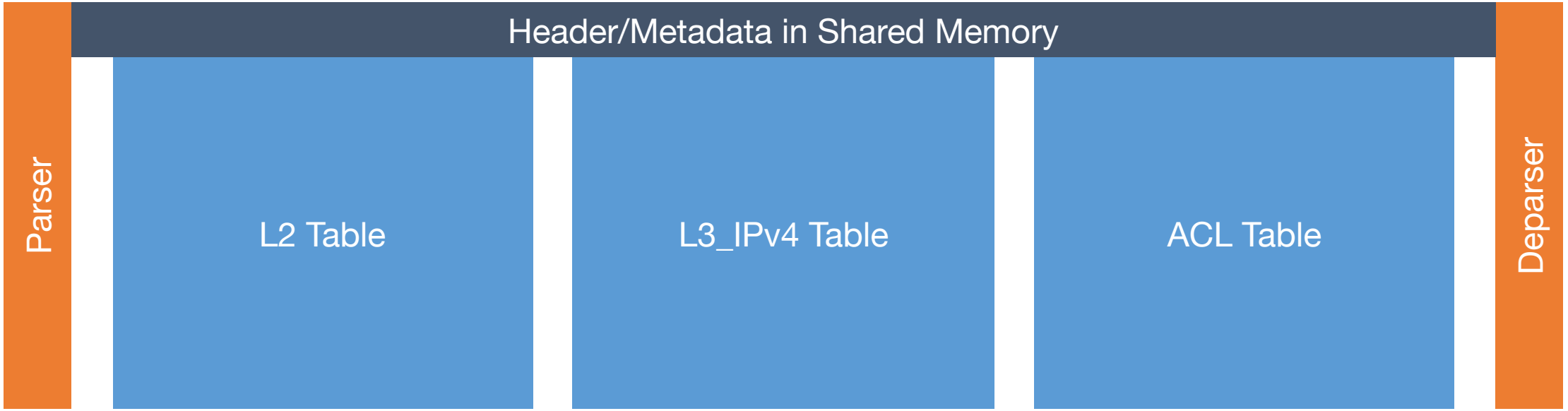


Table graph





Example: Add IPv6 Support

Parser graph

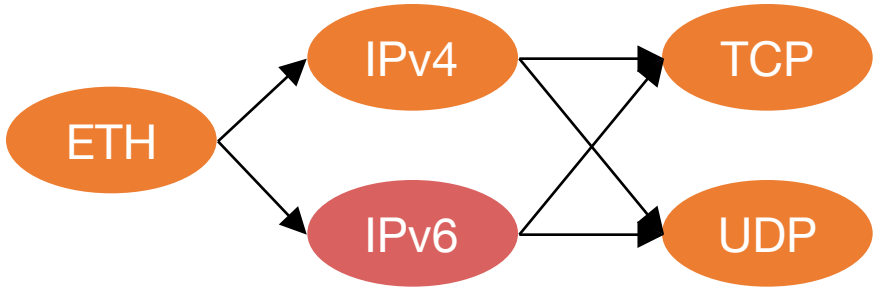
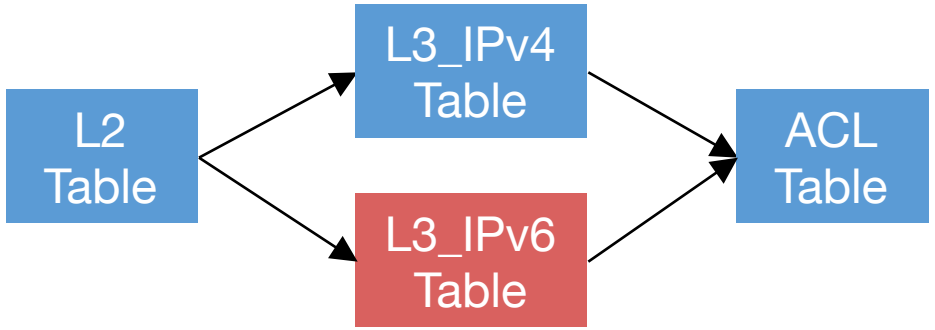
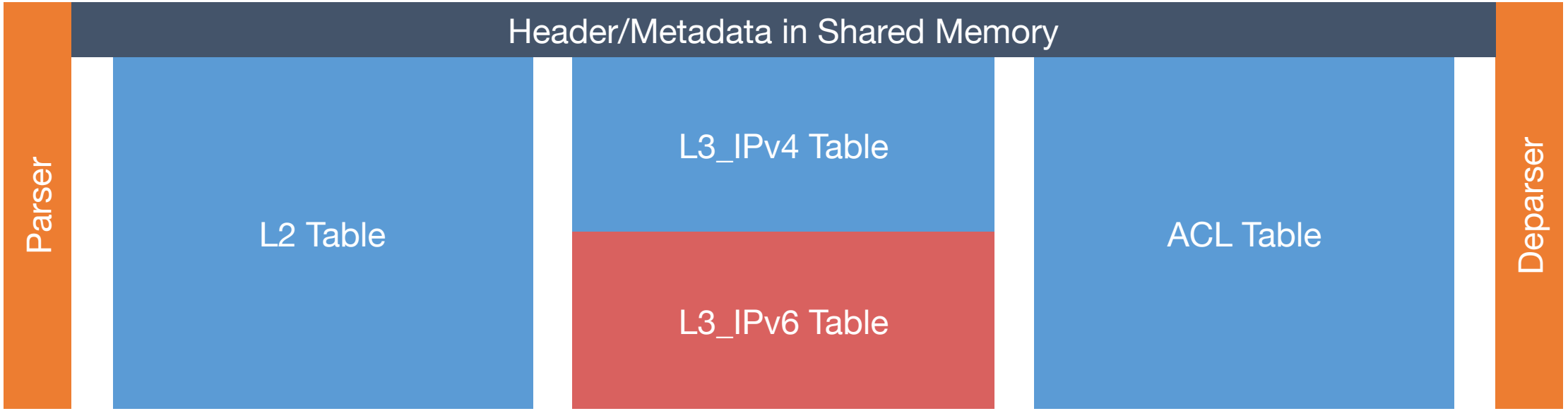
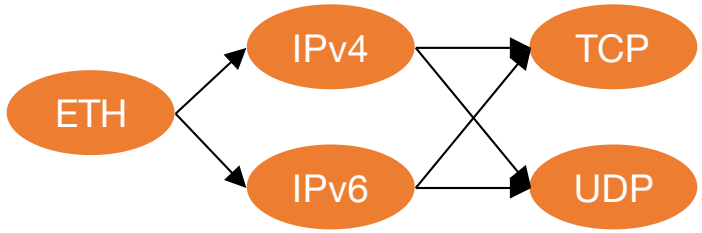


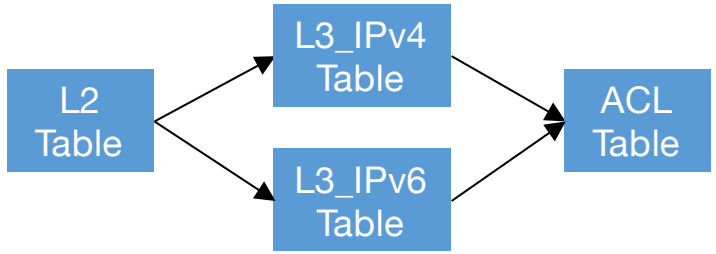
Table graph





P4 Workflow





- You need to write

my_switch.p4

Define your switch

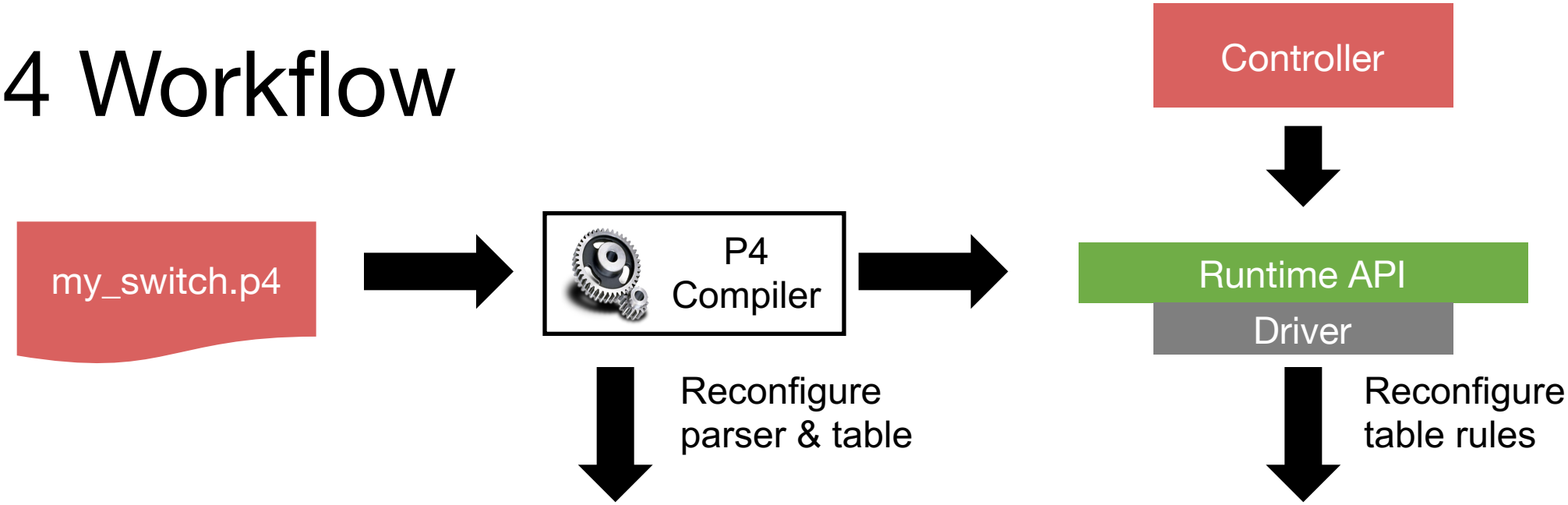
- Specify header format
- Specify parser graph
- Specify type (match & action) of each table
- Specify table graph

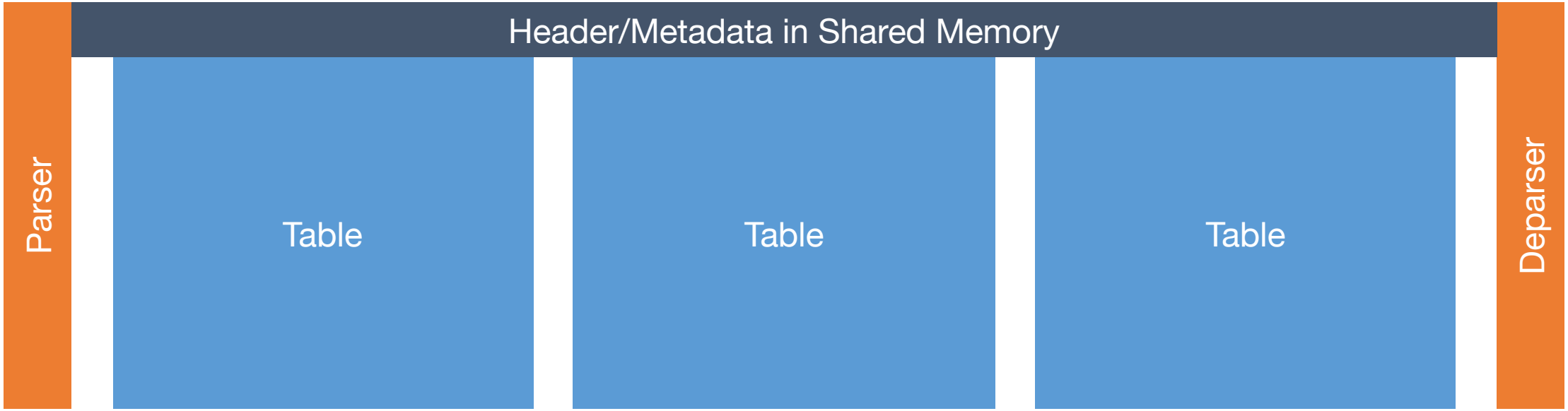
Controller

Define how to operate your switch

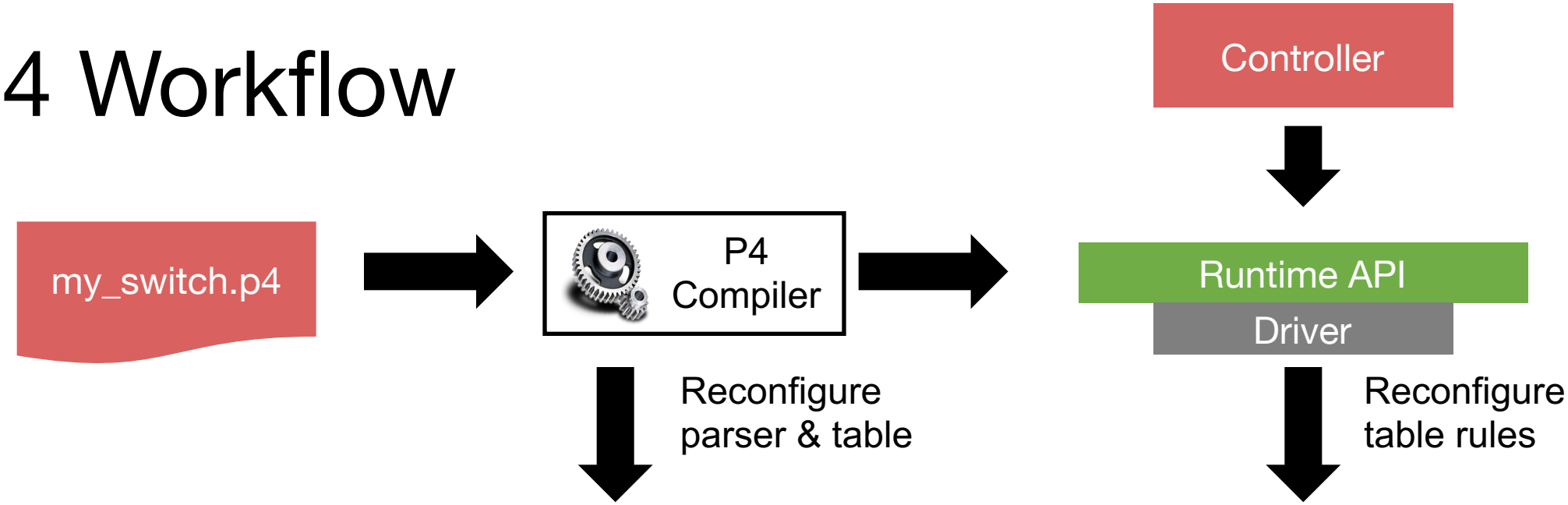
- Collect traffic stats from switch
- Make control decisions
- Reconfigure rules for each table for each switch

P4 Workflow


Controller
my_switch.p4
P4
Compiler
Runtime API
Driver
Reconfigure
parser & table
Reconfigure
table rules


Header/Metadata in Shared Memory
Parser
Table
Table
Table
Deparser

P4 Workflow

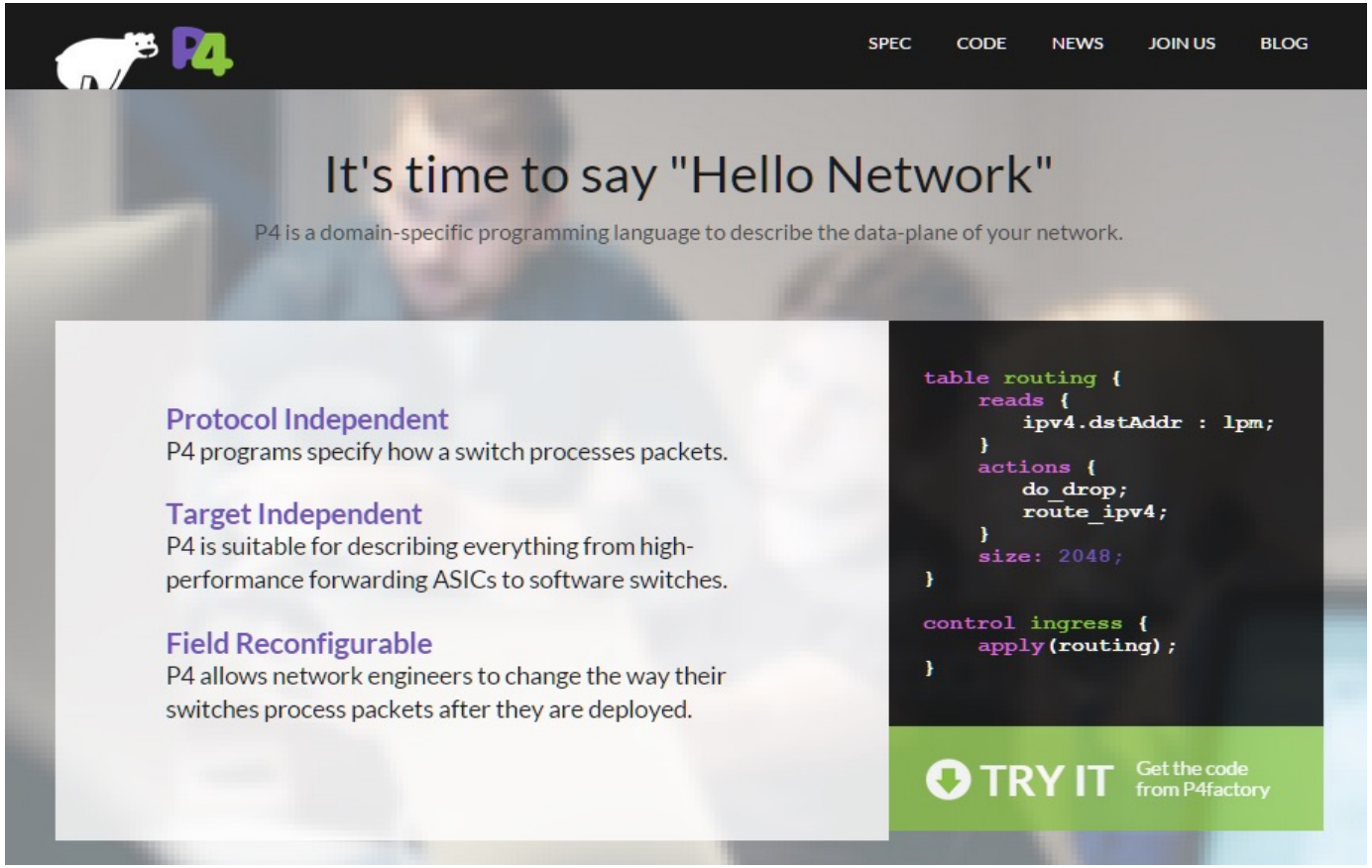


P4 Target

- ASIC
  - Intel Flexpipe, Cisco Doppler, Cavium (Xpliant), Barefoot Tofino, …
- NPU
  - EZchip, Netronome, …
- CPU (Software Switch, NIC)
  - Open Vswitch, eBPF, DPDK, VPP…
- FPGA
  - Xilinx, Altera, …

The P4 Language Consortium (http://p4.org)

- Consortium of academic and industry members
- Open source, evolving, domain-specific language
- Permissive Apache license, code on GitHub today
- Membership is free: contributions are welcome
- Independent, set up as a California nonprofit



P4.org Membership

Original P4 Paper Authors: Barefoot Networks, Google, Intel, Microsoft, Princeton University, Stanford University

Operators: Alibaba Group, at&t, Baidu, Comcast, kt, Microsoft, SK telecom, Tencent

Systems: Brocade, Cisco, Corsa, Dell, Hewlett Packard Enterprise, Huawei, Inventec, Juniper Networks

Targets: Aeponyx, Barefoot Networks, Cavium, EZchip, HGTech, Atomic Rules, Broadcom, centec networks, freescale, intel, Marvell, Netronome, PLUMgrid, Xilinx, Mellanox, MoSys, NoviFlow, vmware

Academia: Cornell University, Politecnico Milano 1863, Princeton University, Stanford University, Université du Luxembourg, Università della Svizzera italiana

- **Open source**, evolving, domain-specific language
- Permissive Apache license, code on GitHub today
- **Membership is free**: contributions are welcome
- Independent, set up as a California nonprofit

Let’s write your first switch in P4…

P4 Workflow

- Your need to write

my_switch.p4

Define your switch

- Specify header format
- Specify parser graph
- Specify type (match & action) of each table
- Specify table graph

Controller

Define how to operate your switch

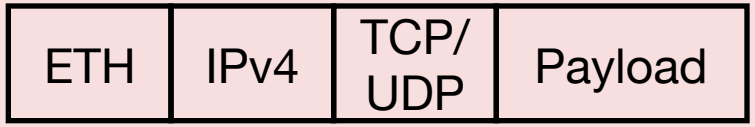
- Collect traffic stats from switch
- Make control decisions
- Reconfigure rules for each table for each switch

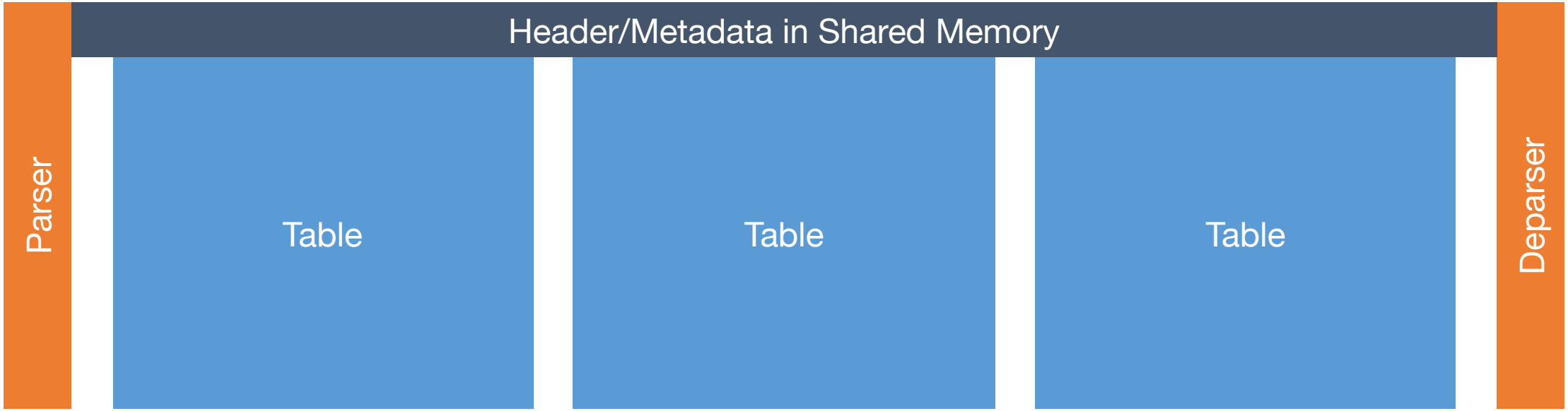
Write P4 Switch Specification

Header Declaration

my_switch.p4

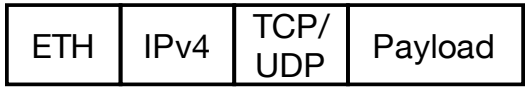
Packet format





Write P4 Switch Specification

Packet format

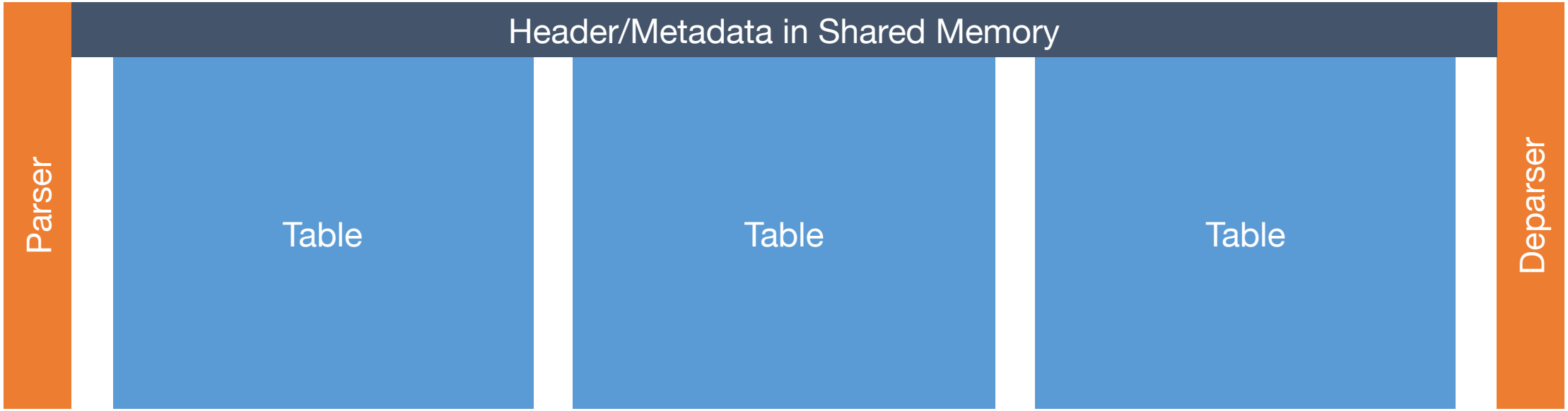


Header Declaration

my_switch.p4

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

```
header_type ip4_t {…}
header ipv4_t ipv4;

header_type tcp_t {…}
header tcp_t tcp;

header_type udp_t {…}
header udp_t udp4;
```



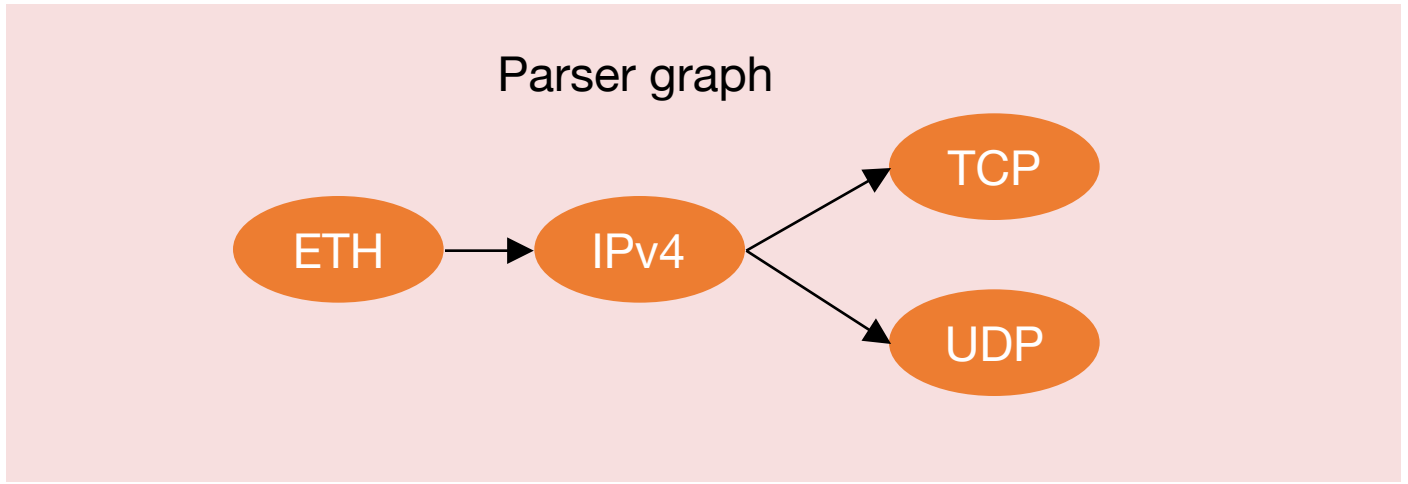
Write P4 Switch Specification

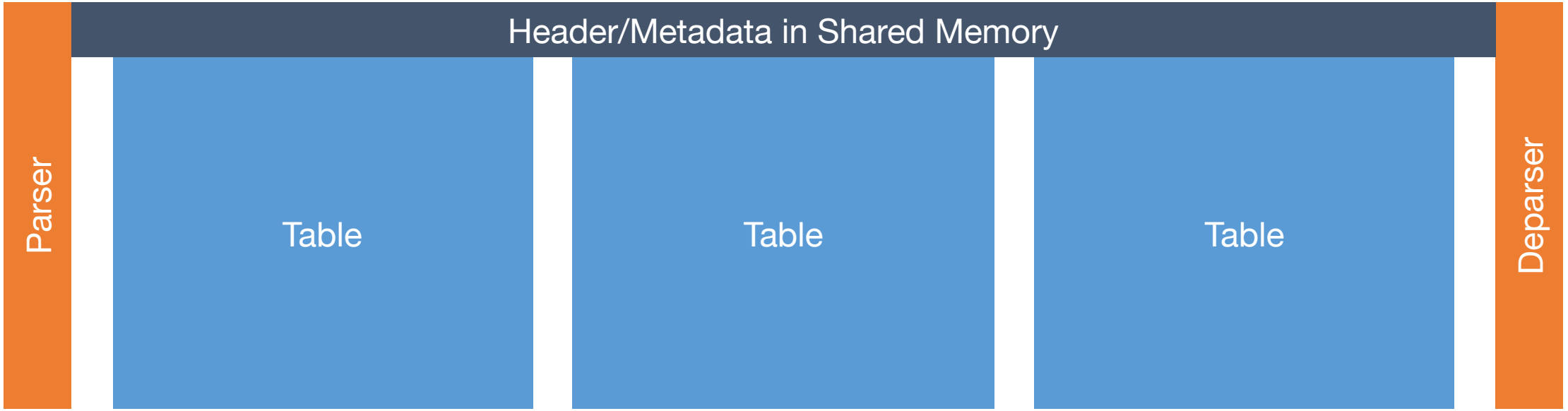
```
header_type ethernet_t {
    fields {
        dstAddr:   48;
        srcAddr:   48;
        etherType: 16;
    }
}
header ethernet_t ethernet;
```

```
header_type ipv4_t {...}
header ipv4_t ipv4;
header_type tcp_t {...}
header tcp_t tcp;
header_type udp_t {...}
header udp_t udp;
```

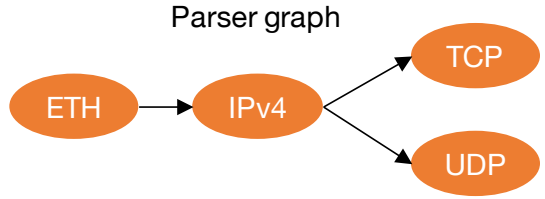
Parser Graph Declaration

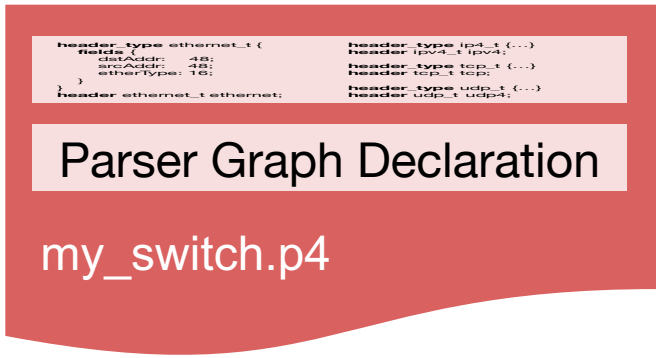
my_switch.p4





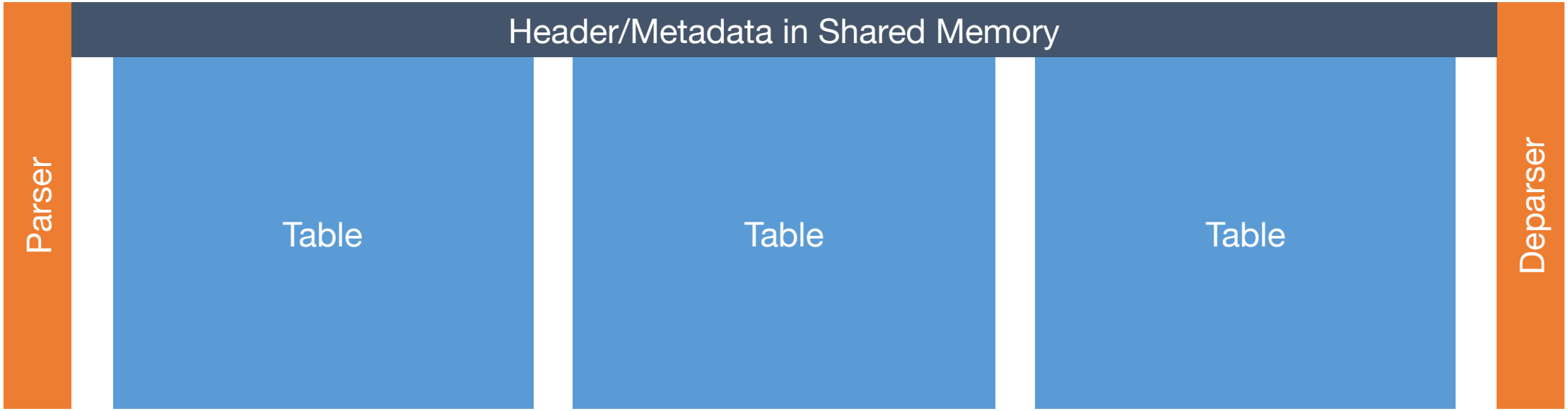
Write P4 Switch Specification



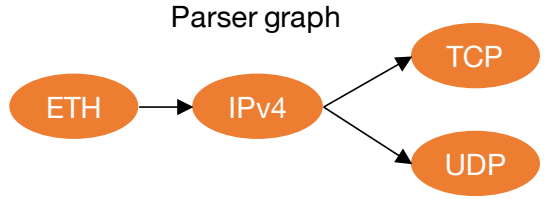



```
parser start {
    extract(ethernet);
    return select(latest.etherType) {
        0x0800: parse_ipv4;
        default: ingress;
}
```

```
parser parse_ipv4 {
    extract(ipv4);
    return select(latest.protocol) {
        6:      parse_tcp;
        17:     parse_udp;
        default: ingress;
}
parser parse_tcp  {…}
parser parse_udp {…}
```

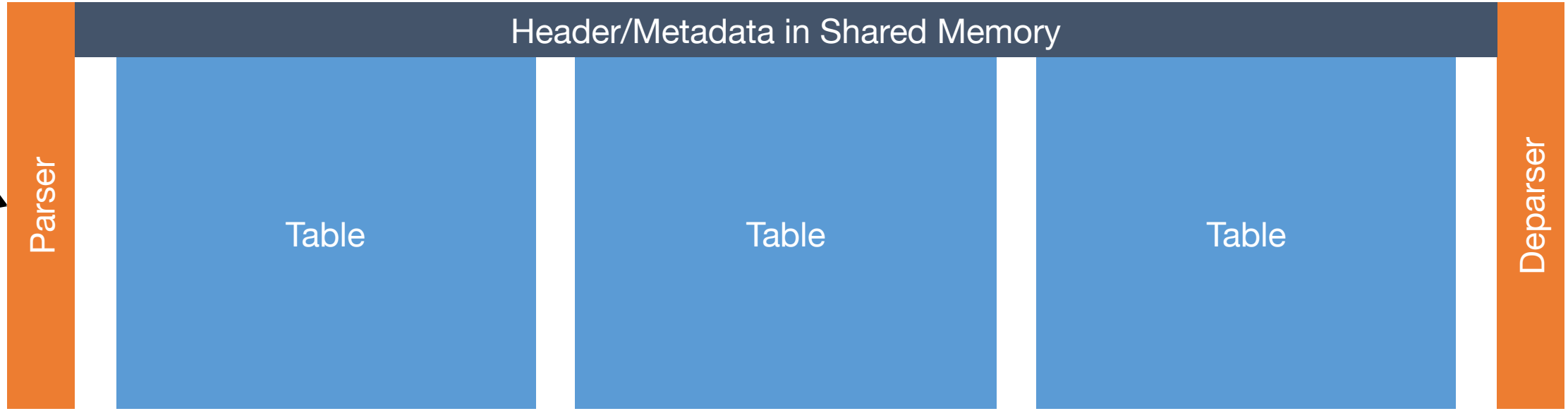


Write P4 Switch Specification

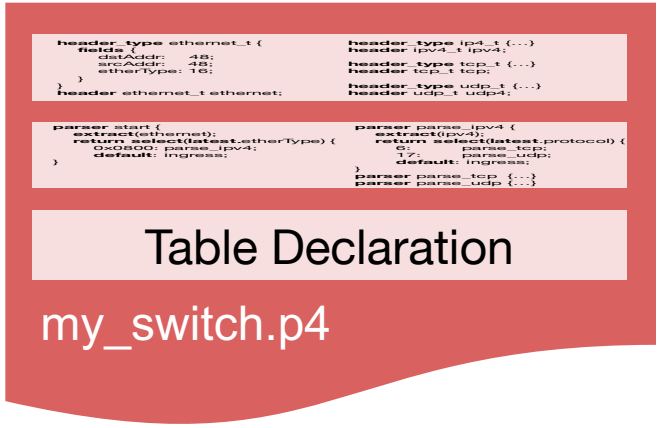



```
parser start {
    extract(ethernet);
    return select(latest.etherType) {
        0x0800: parse_ipv4;
        default: ingress;
}
```

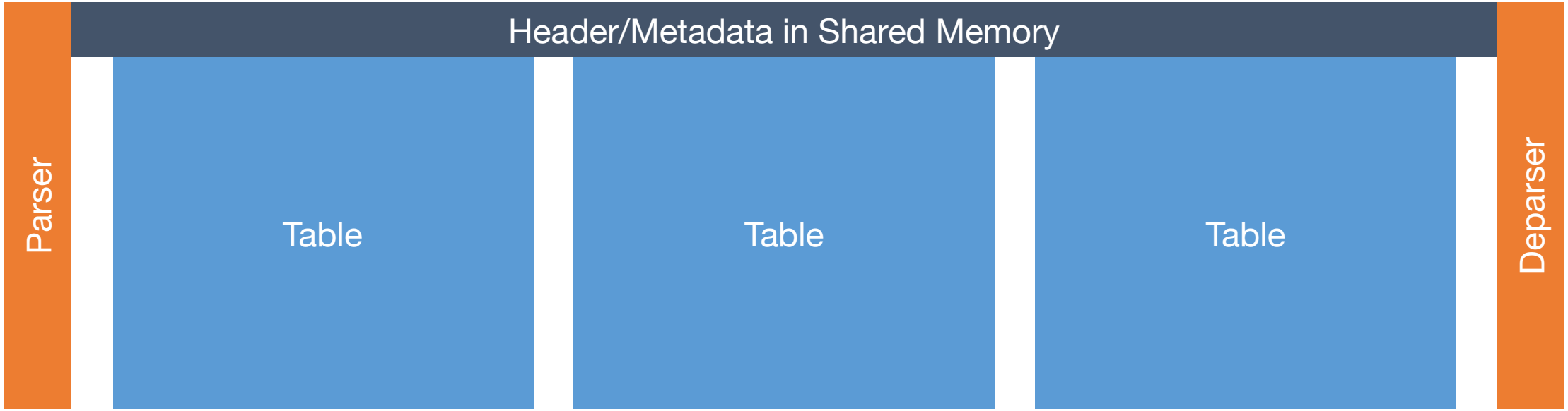
```
parser parse_ipv4 {
    extract(ipv4);
    return select(latest.protocol) {
        6:          parse_tcp;
        17:         parse_udp;
        default: ingress;
}
parser parse_tcp  {…}
parser parse_udp {…}
```



Write P4 Switch Specification




```
table L2 {
    reads {
        eth.dstAddr: exact;
    }
    actions {
        set_port;
        no_op;
    }
    size: 4096 }
```

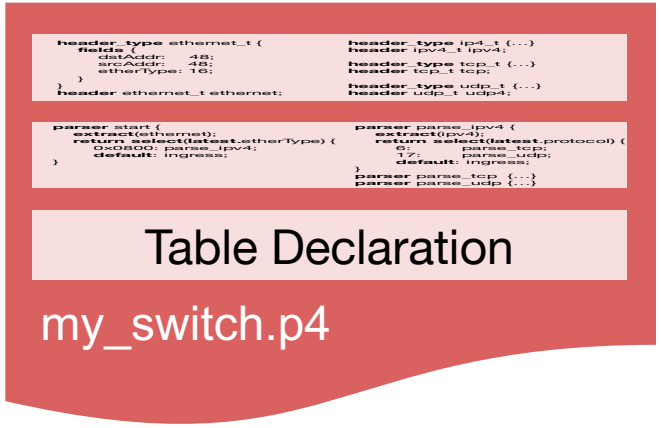


Actions

- Primitive actions
  - no_op, drop
  - modify_field, modify_field_with_hash_based_offset
  - add, add_to_field
  - register_read, register_write
  - add_header, remove_header, copy_header, push/pop (a header)
  - count, execute_meter
  - resubmit, recirculate, clone
  - etc.
- Compound actions

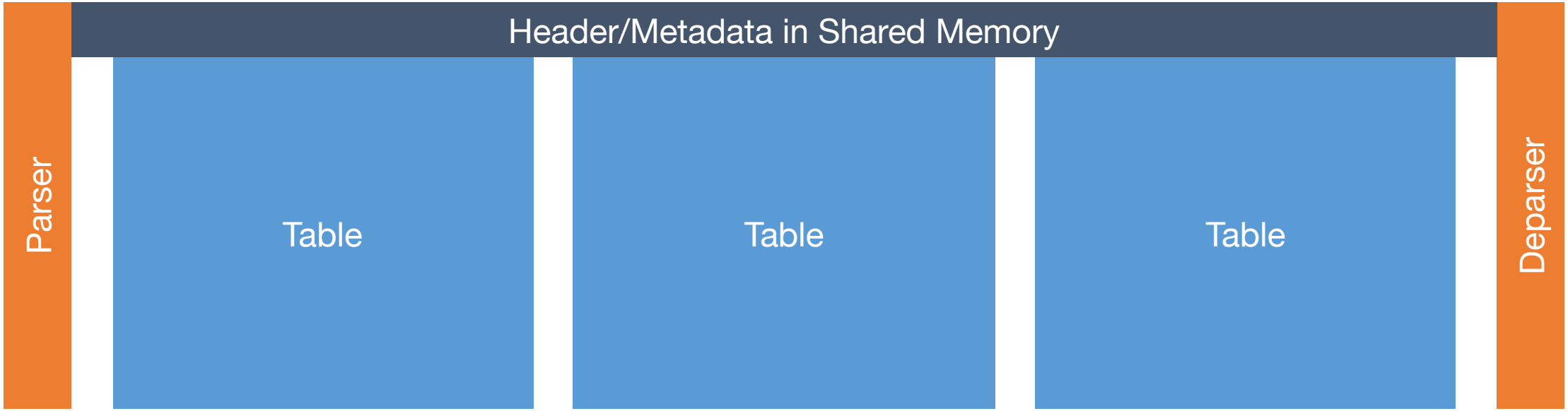
```
action compound_action(dst_port, dst_ip) {
    modify_field(standard_metadata.egress_spec, dst_port)
    modify_field(ipv4.dst_ip, dst_ip);
    add_to_field(ipv4.ttl, -1)
}
```

Write P4 Switch Specification

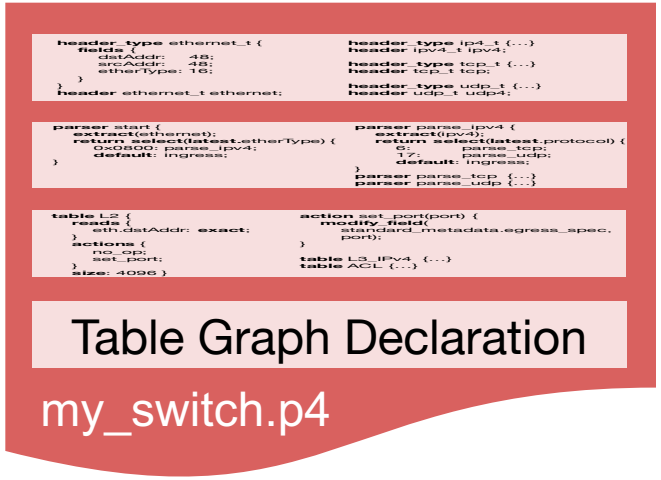


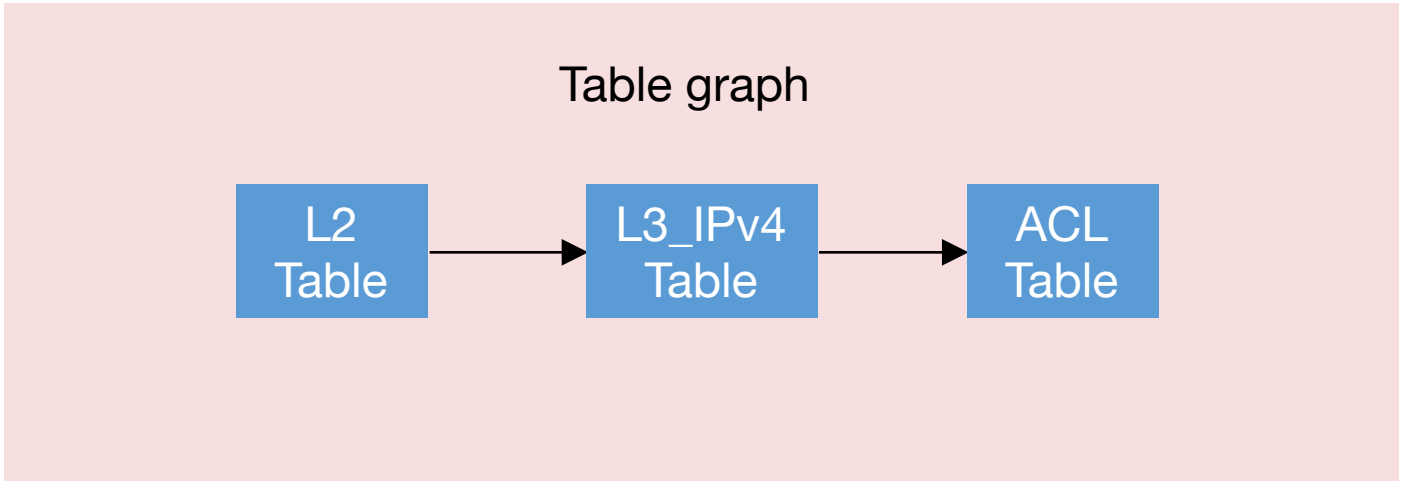

```
table L2 {
    reads {
        eth.dstAddr: exact;
    }
    actions {
        no_op;
        set_port;
    }
    size: 4096 }
```

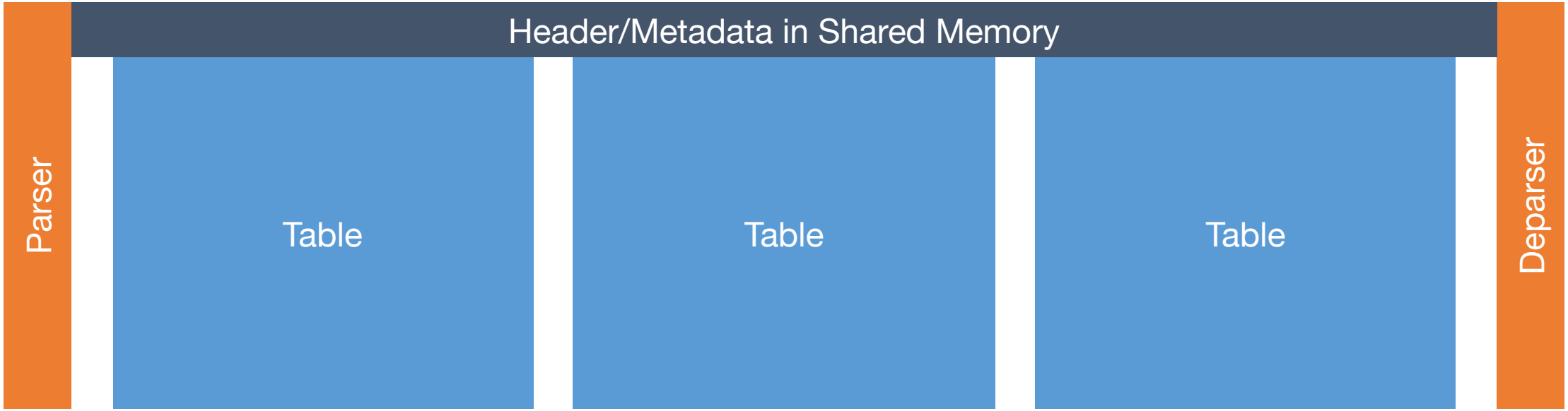
```
action set_port(port) {
    modify_field(
        standard_metadata.egress_spec,
        port);
}

table L3_IPv4  {…}
table ACL {…}
```



Write P4 Switch Specification

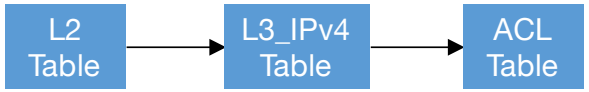


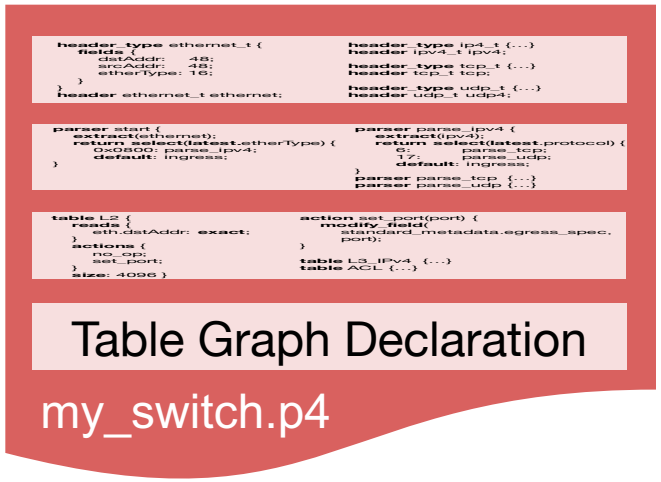




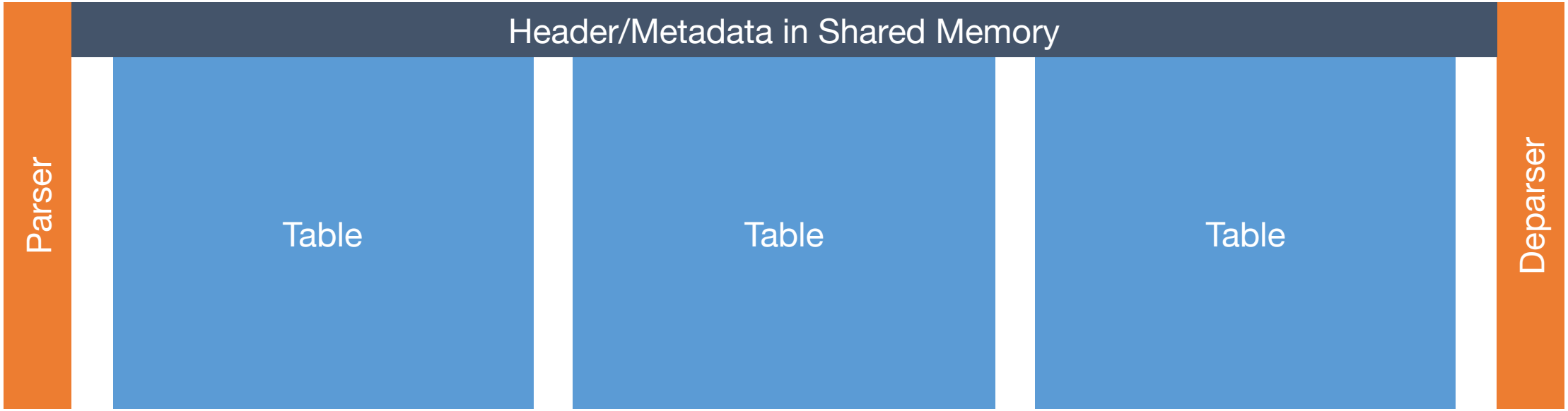
Write P4 Switch Specification

Table graph



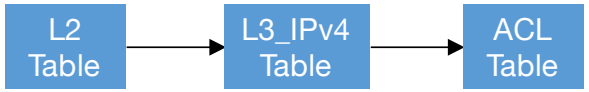


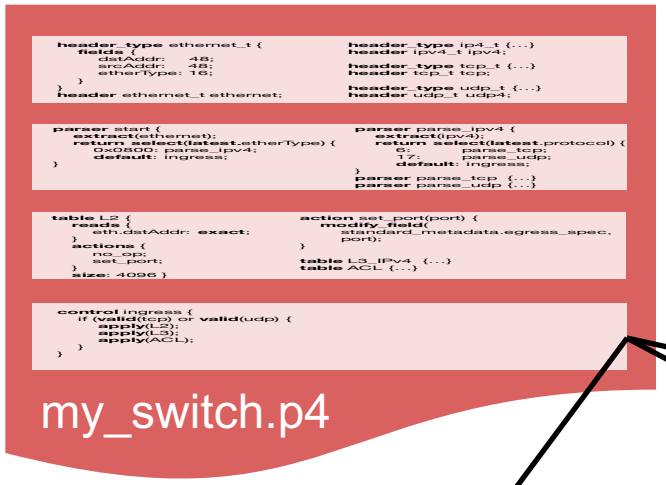

```
control ingress {
    if (valid(tcp) or valid(udp) {
        apply(L2);
        apply(L3);
        apply(ACL);
    }
}
```



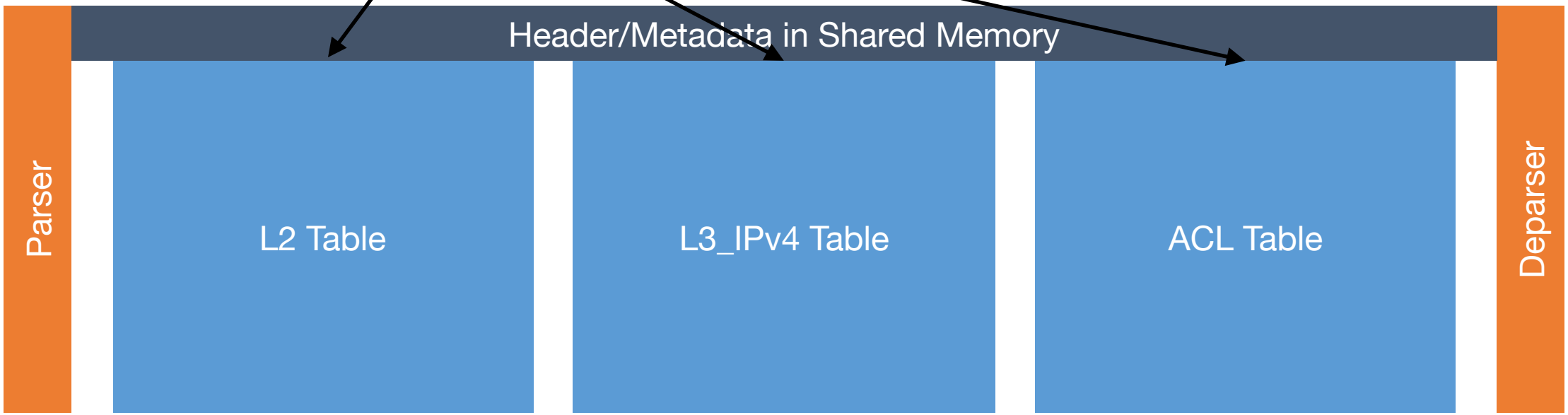
Write P4 Switch Specification

Table graph

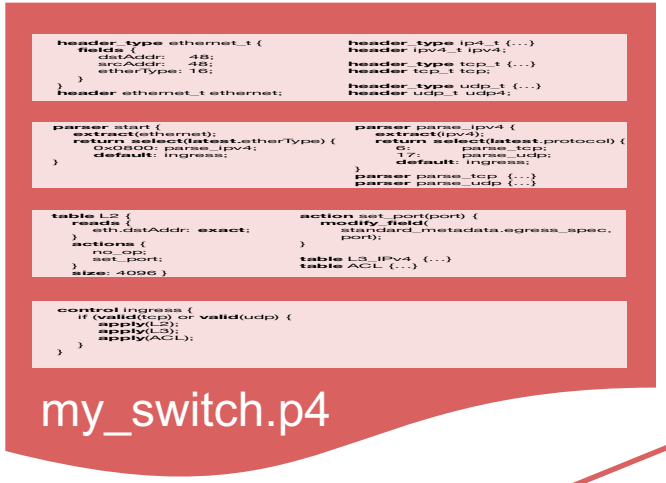





```
control ingress {
    if (valid(tcp) or valid(udp) {
        apply(L2);
        apply(L3);
        apply(ACL);
    }
}
```



Write Controller to Install Rules



Controller

Parser

Header/Metadata in Shared Memory

Match	Action
dst_mac= 00:A0:C9:14:C8:29	out_port=2
dst_mac= AC:FF:92:32:A7:64	out_port=1
dst_mac= F7:6A:72:7B:CC:43	out_port=7
default	no_op

Match	Action
dst_ip= 10.0.0.0/24	no_op
dst_ip= 10.0.1.0/24	out_port=2
dst_ip= 10.0.0.0/16	out_port=5
default	drop

Match	Action
protocol=6(tcp), dst_port=80(http)	pass
protocol=6(tcp), dst_port=22(ssh)	drop
src_ip=12.0.0.0/24, dst_ip=24.12.54.2/32	drop
default	drop

Deparser

Thanks!