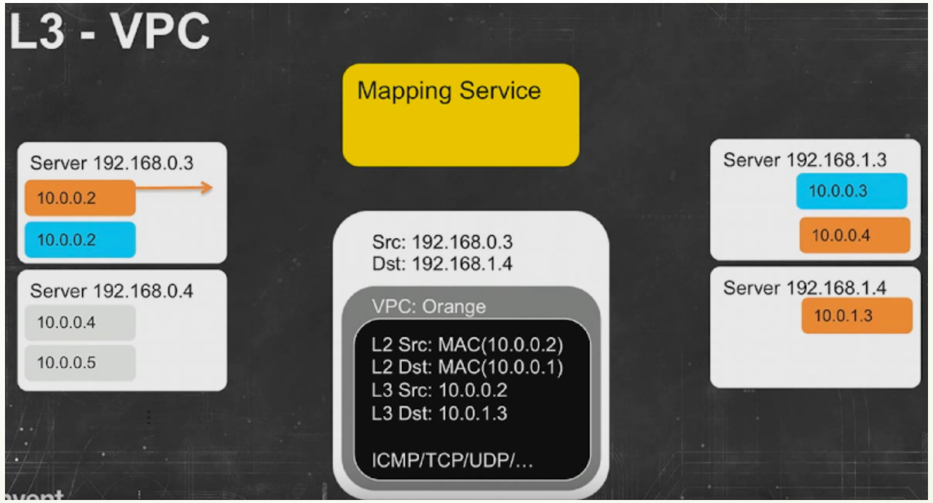
P4
Virtual Private Cloud

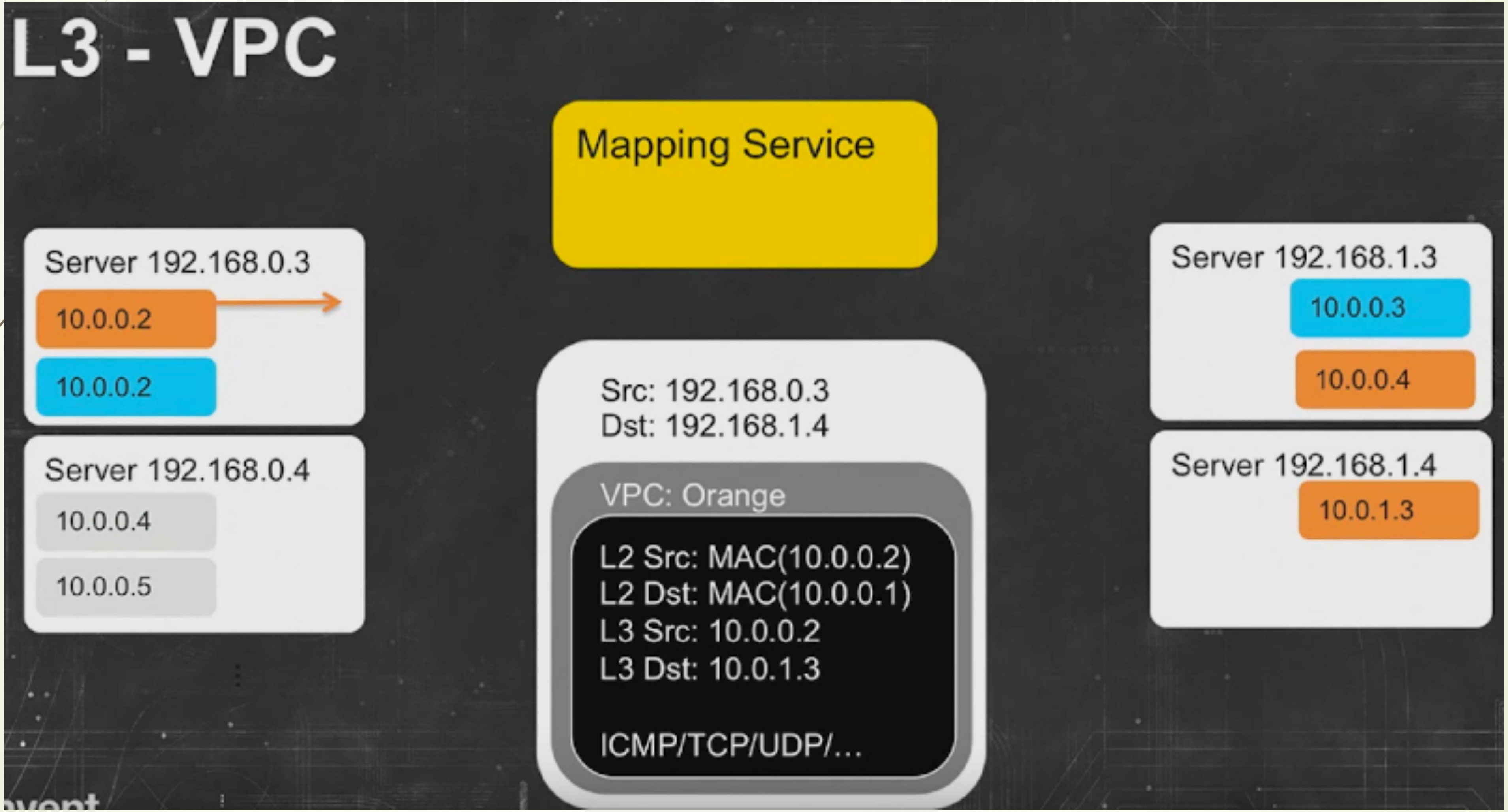
Based on Amazon AWS VPC presentation
https://www.youtube.com/watch?v=Zd5hsL-JNY4
https://www.youtube.com/watch?v=3qln2u1Vr2E

Amazon Virtual Private Cloud

- Provide multitenancy in the same hardware infrastructure
- AWS has developed their own encapsulation mechanism
- Centralized mapping service with distributed cache on each compute.
- Each compute cache should have 100% of their required rules.



Amazon Virtual Private Cloud



L3 - VPC

Mapping Service

Server 192.168.0.3
- 10.0.0.2
- 10.0.0.2

Server 192.168.0.4
- 10.0.0.4
- 10.0.0.5

Src: 192.168.0.3
Dst: 192.168.1.4

VPC: Orange

L2 Src: MAC(10.0.0.2)
L2 Dst: MAC(10.0.0.1)
L3 Src: 10.0.0.2
L3 Dst: 10.0.1.3

ICMP/TCP/UDP/...

Server 192.168.1.3
- 10.0.0.3
- 10.0.0.4

Server 192.168.1.4
- 10.0.1.3

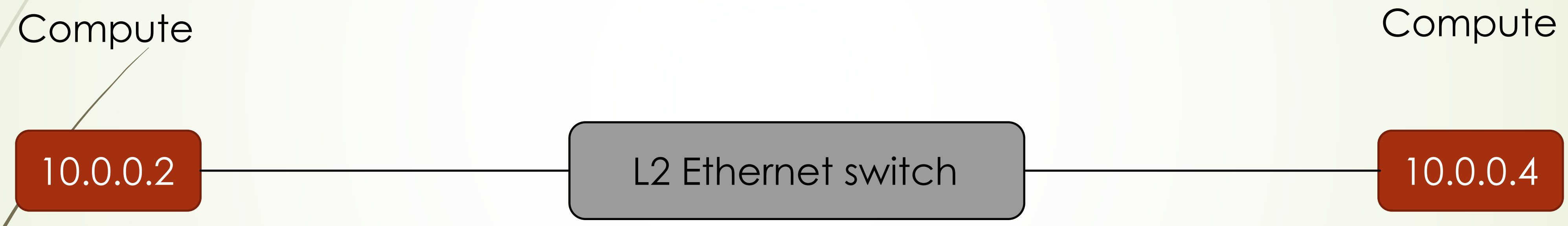
P4
Protocol-Independent Packet Processor

- Header
  - definition of the packet structures
- Parsers
  - Parse the packet based on known headers and triggers the entry p
- Control
  - Contains the logic of which tables to be executed
- Tables
  - Match and apply actions
- Actions
  - Packet manipulation

P4 Private Virtual Cloud

- ARP requests are only replied by P4 Switches
  - Capture ARP request
  - Convert the ARP request into a ARP reply
- IP packets are encapsulated with a custom and non standard protocol called VPC
  - Ethernet type 0x0777
  - Header: Customer id, src/dst IP , src/dst P4 switch
- VPC packets are forwarded by intermediate switches using dst P4 switch label
- Egress switches will remove VPC encapsulation and deliver the IP packet to the node
  - If hosts belong to different networks, src/dst mac addresses will be overwritten
- Non ARP or IP packets are dropped
- ARP or IP packets that does not meet the criteria will be dropped too. For example, an unknown hosts

L2 – Ethernet (standard)



Compute

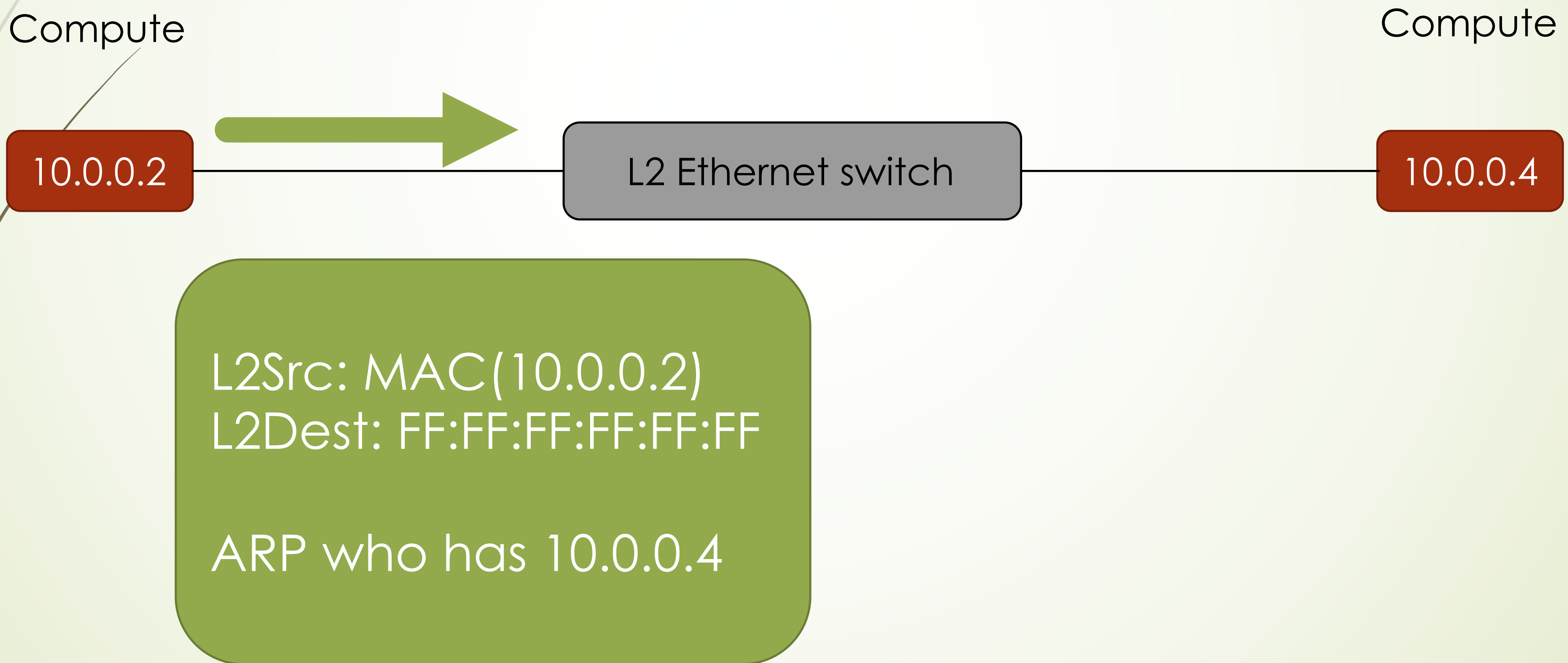
10.0.0.2

L2 Ethernet switch

Compute

10.0.0.4

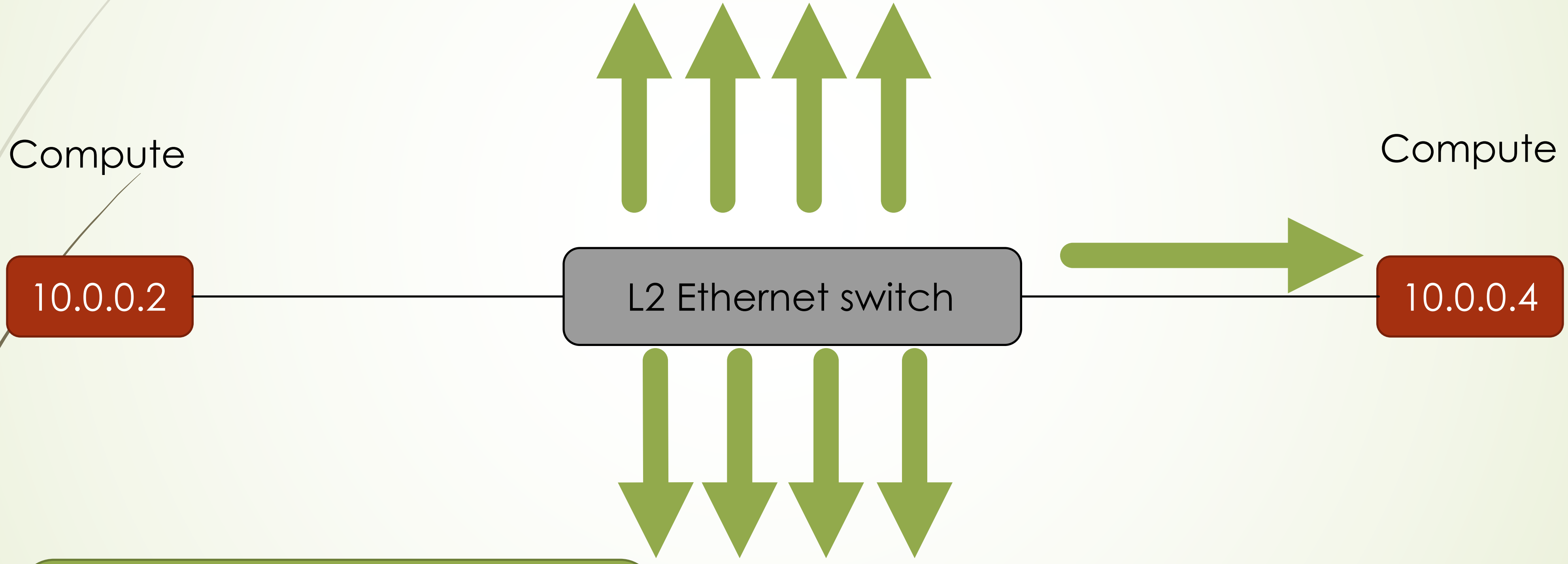
L2 – Ethernet (standard)



L2Src: MAC(10.0.0.2)
L2Dest: FF:FF:FF:FF:FF:FF

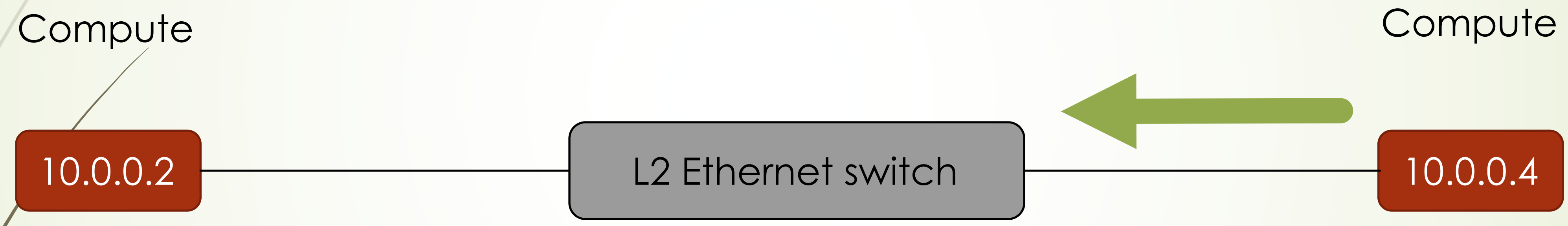
ARP who has 10.0.0.4

L2 – Ethernet (standard)



Packet is flooded

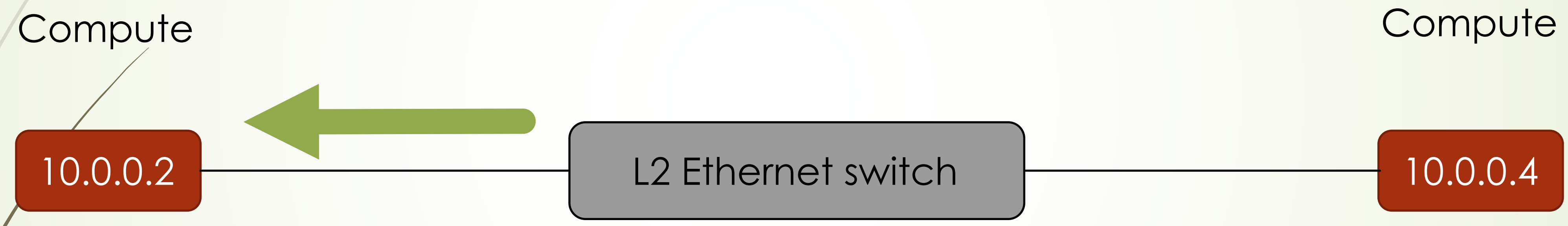
L2 – Ethernet (standard)



L2Src: MAC(10.0.0.4)
L2Dest: MAC(10.0.0.2)

ARP I have 10.0.0.4

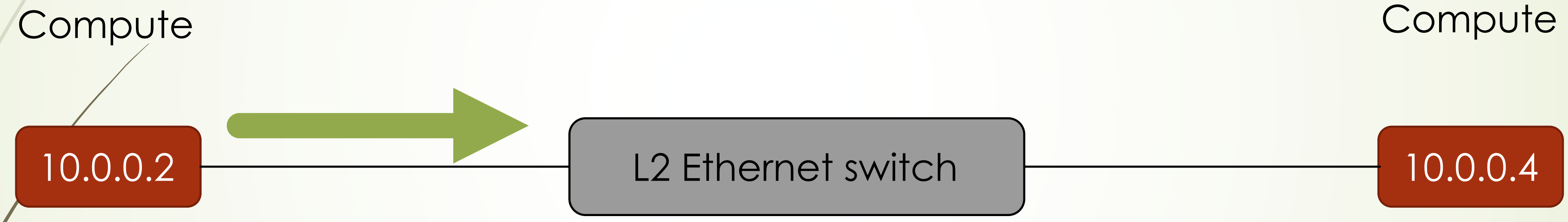
L2 – Ethernet (standard)



L2Src: MAC(10.0.0.4)
L2Dest: MAC(10.0.0.2)

ARP I have 10.0.0.4

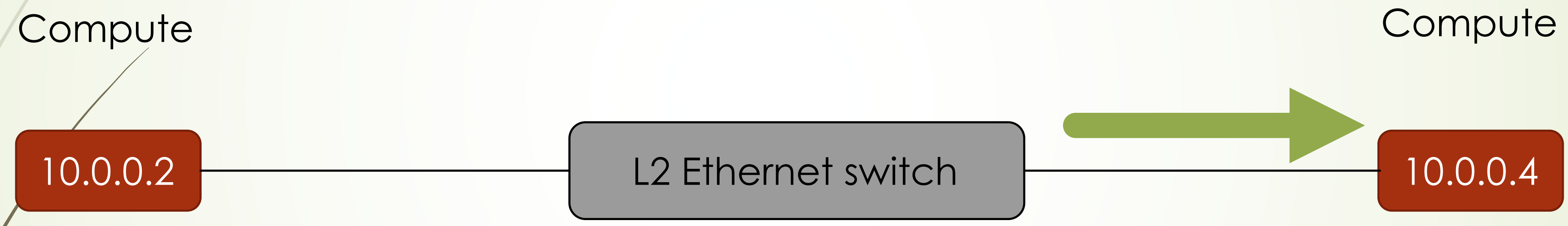
L2 – Ethernet (standard)



L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.4)

ICMP answer to10.0.2

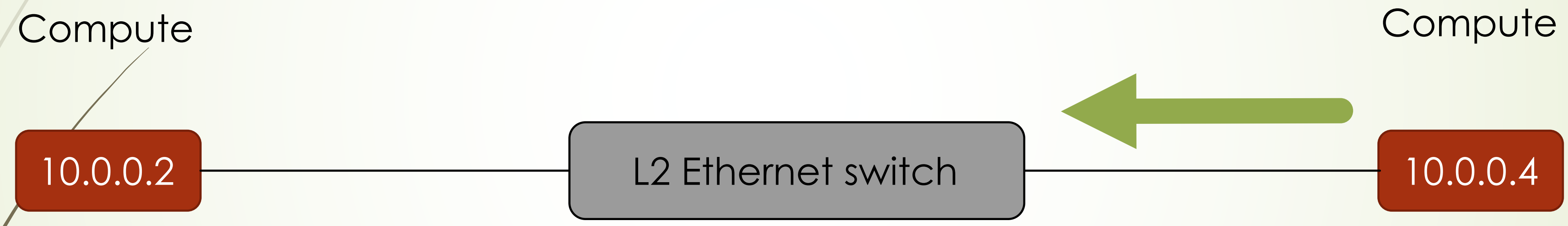
L2 – Ethernet (standard)



L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.4)

ICMP answer to10.0.2

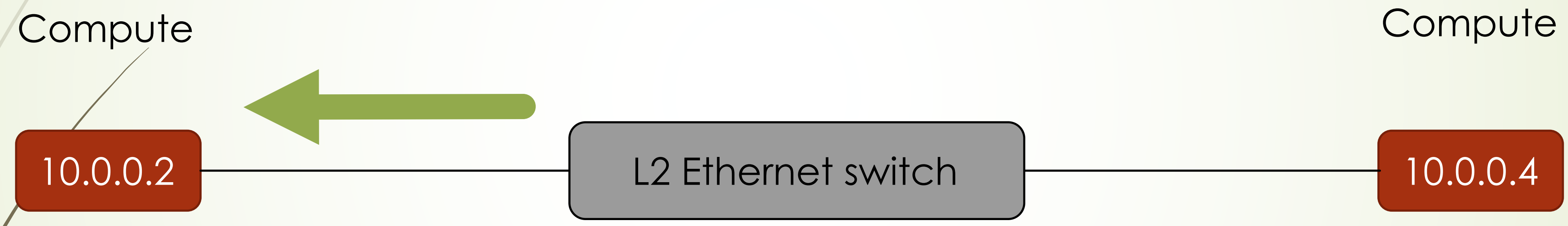
L2 – Ethernet (standard)



L2Src: MAC(10.0.0.4)
L2Dest: MAC(10.0.0.2)

ICMP reply from 10.0.0.4

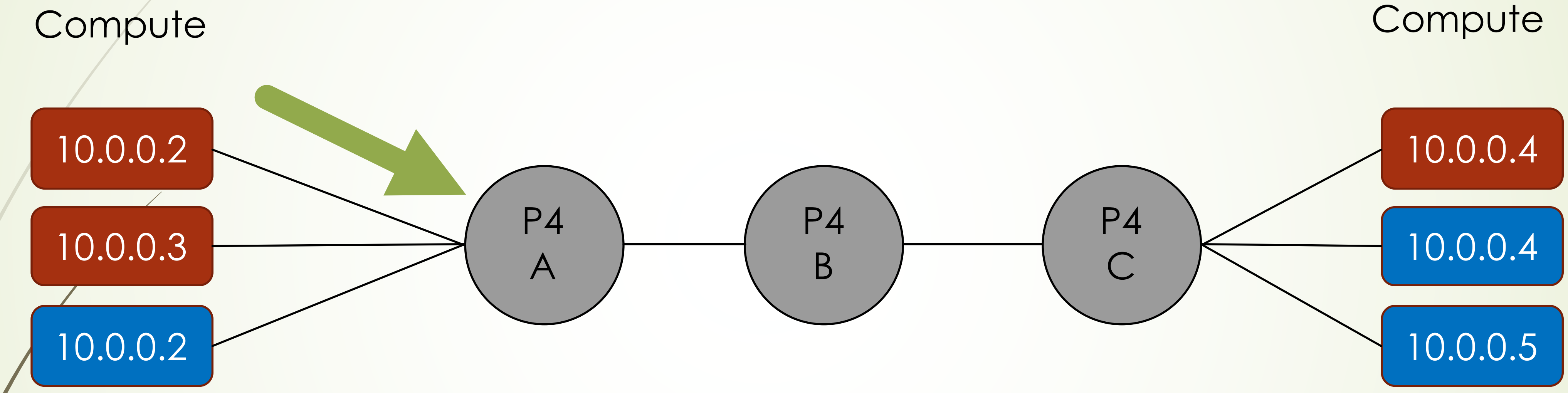
L2 – Ethernet (standard)



L2Src: MAC(10.0.0.4)
L2Dest: MAC(10.0.0.2)

ICMP reply from 10.0.0.4

L2 – Ethernet (using P4 VPC)



L2Src: MAC(10.0.0.2)
L2Dest: FF:FF:FF:FF:FF:FF

ARP who has 10.0.0.4

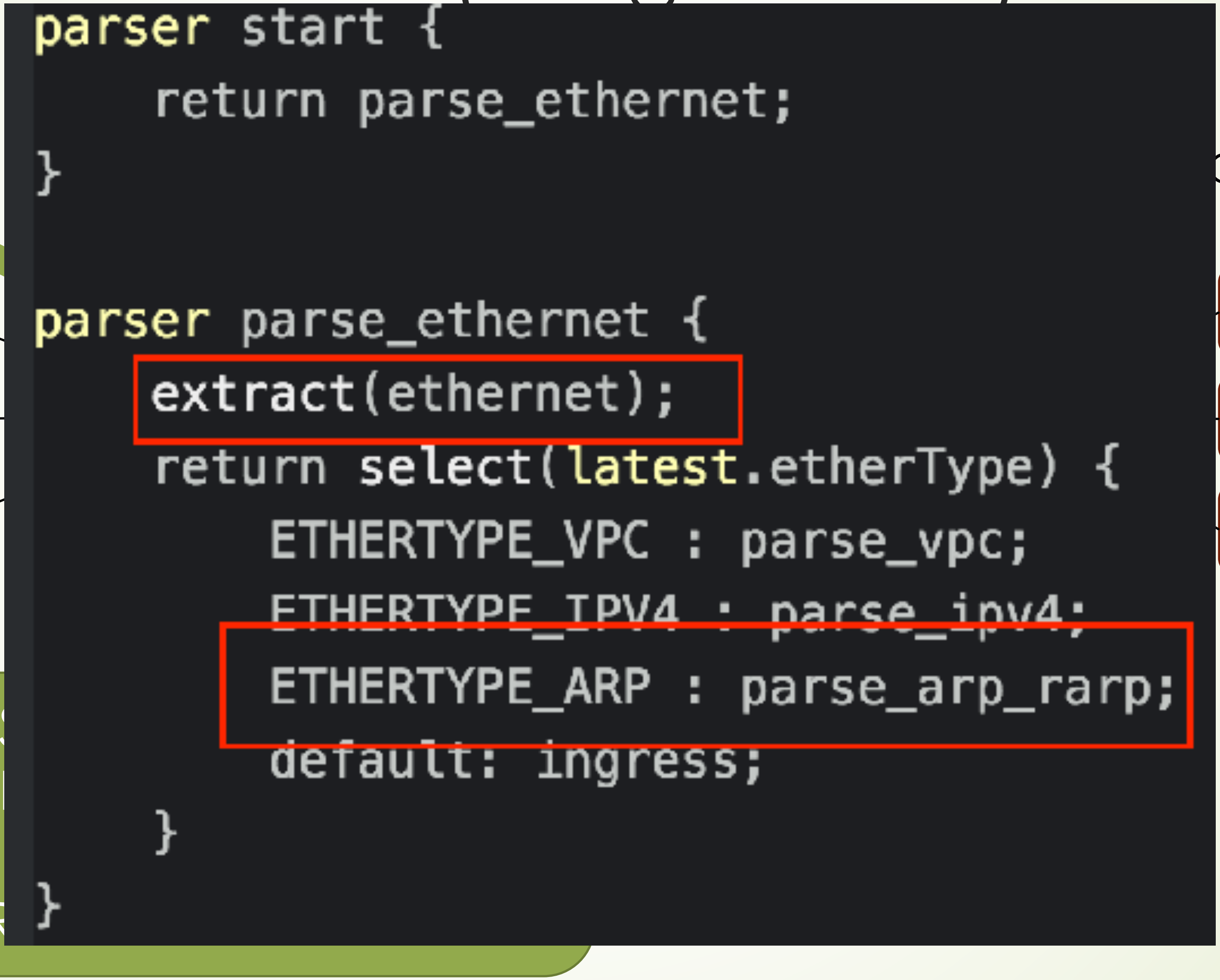
L2 – Ethernet (using P4 VPC)

Compute

10.0.0.2

10.0.0.3

10.0.0.2



```
parser start {
    return parse_ethernet;
}

parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        ETHERTYPE_VPC : parse_vpc;
        ETHERTYPE_IPV4 : parse_ipv4;
        ETHERTYPE_ARP : parse_arp_rarp;
        default: ingress;
    }
}
```

Compute

10.0.0.4

10.0.0.4

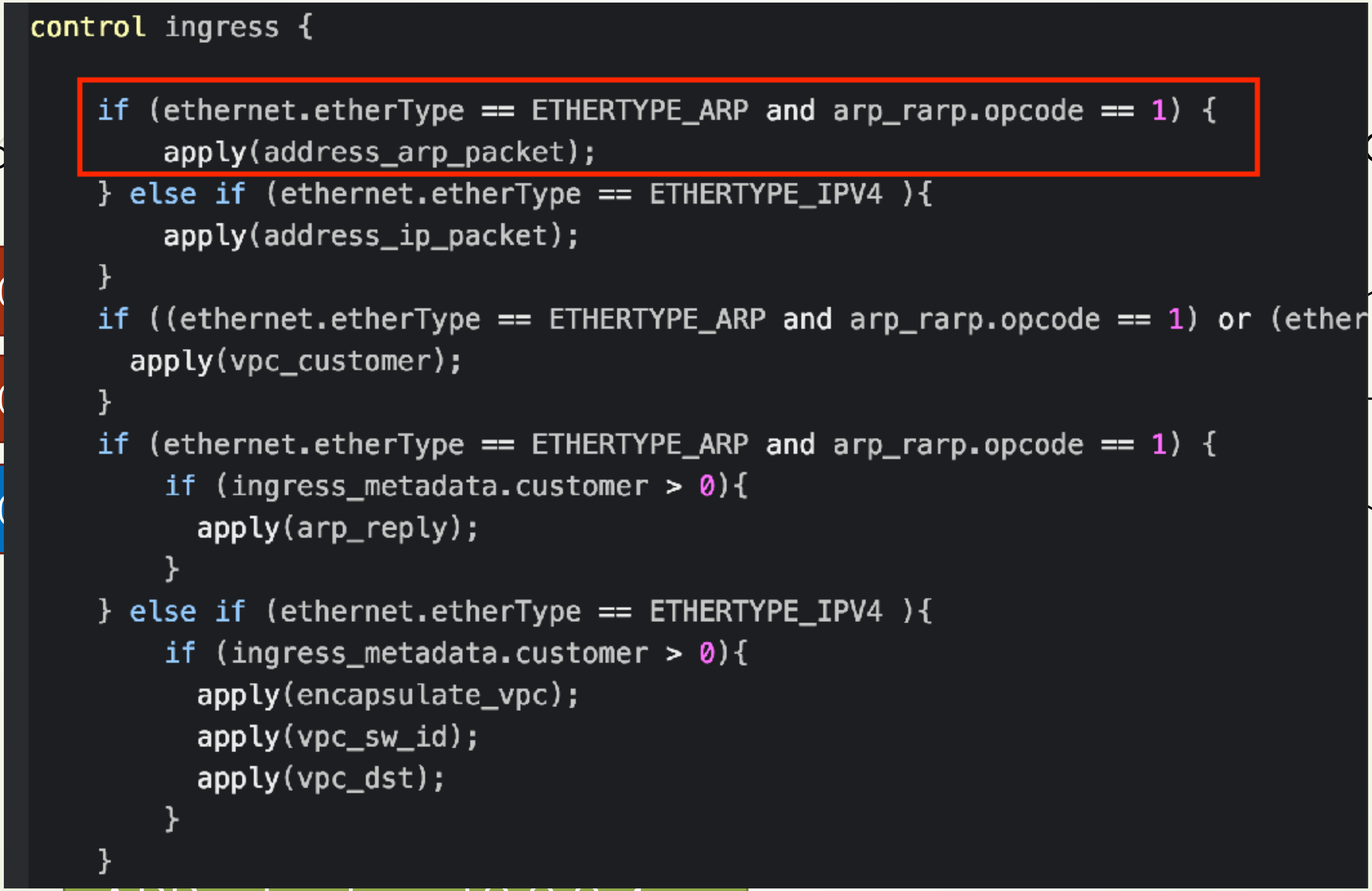
10.0.0.5

L2S

L2I

AR

L2 – Ethernet (using P4 VPC)



```
control ingress {

    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        apply(address_arp_packet);
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        apply(address_ip_packet);
    }
    if ((ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) or (ether
      apply(vpc_customer);
    }
    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        if (ingress_metadata.customer > 0){
          apply(arp_reply);
        }
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        if (ingress_metadata.customer > 0){
          apply(encapsulate_vpc);
          apply(vpc_sw_id);
          apply(vpc_dst);
        }
    }
```

Comp

10.0.(

10.0.(

10.0.(

Compute

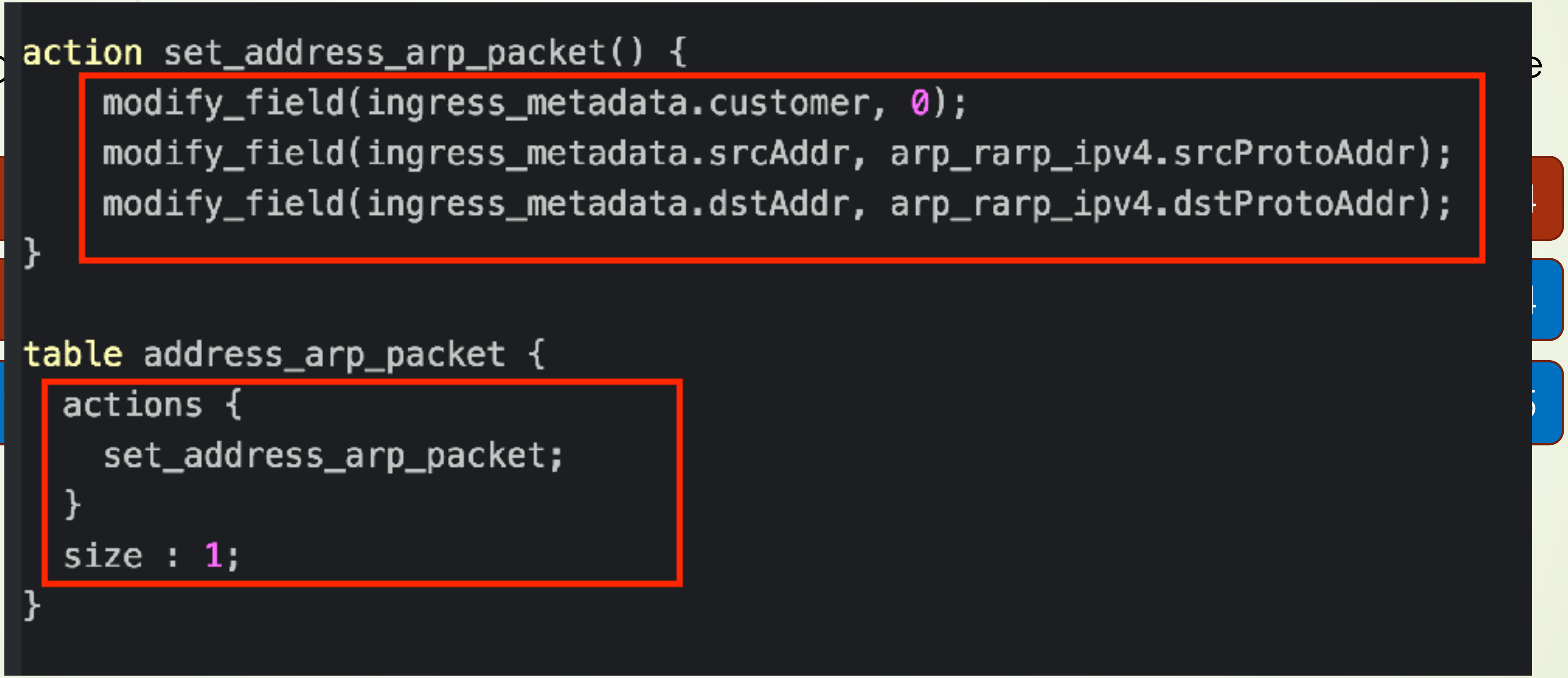
10.0.0.4

10.0.0.4

10.0.0.5

ARP who has 10.0.0.4

L2 – Ethernet (using P4 VPC)



```
action set_address_arp_packet() {
    modify_field(ingress_metadata.customer, 0);
    modify_field(ingress_metadata.srcAddr, arp_rarp_ipv4.srcProtoAddr);
    modify_field(ingress_metadata.dstAddr, arp_rarp_ipv4.dstProtoAddr);
}

table address_arp_packet {
  actions {
    set_address_arp_packet;
  }
  size : 1;
}
```

L2 – Ethernet (using P4 VPC)

Compute

10.0.0.2

10.0.0.3

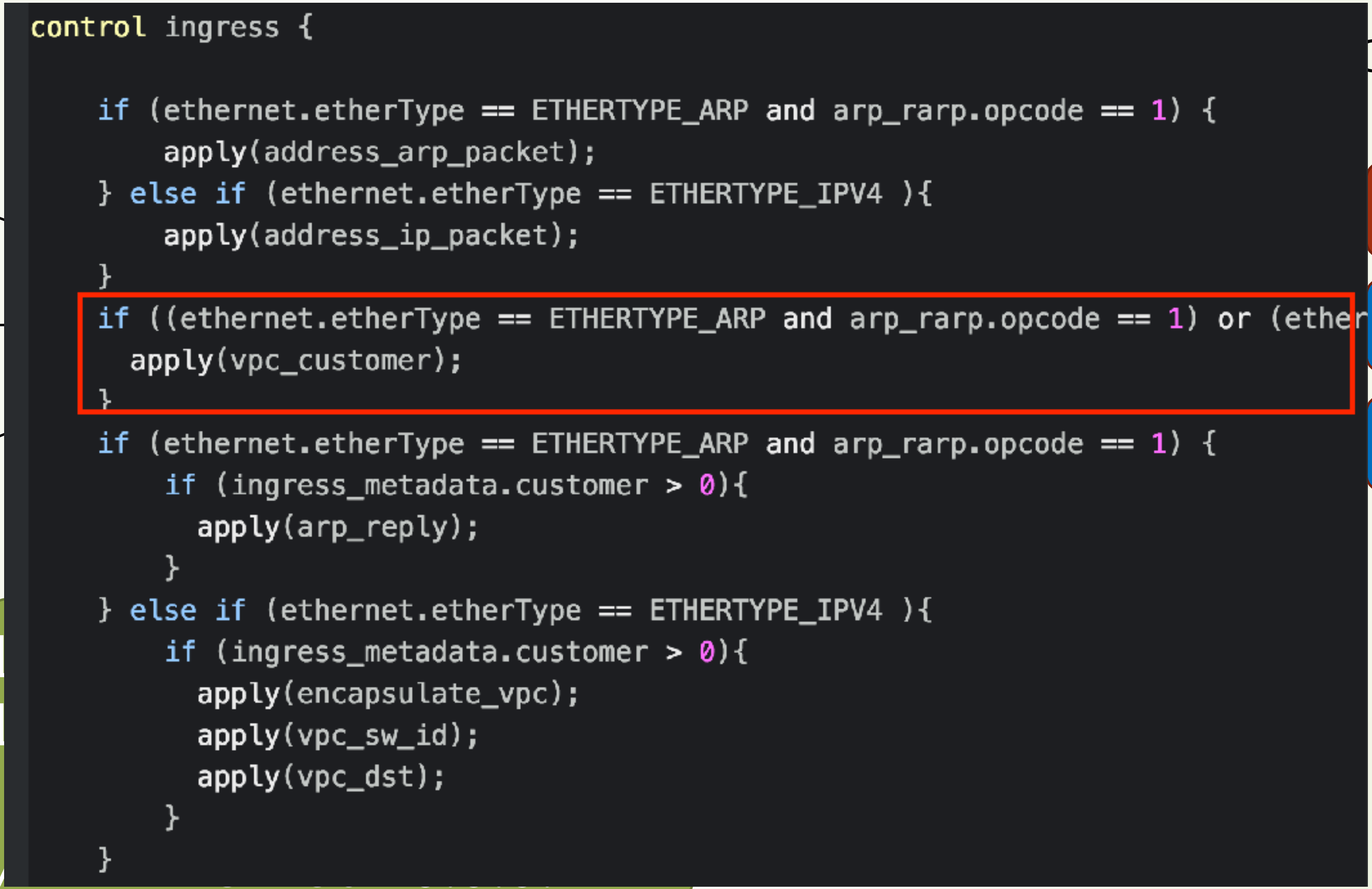
10.0.0.2

Compute

10.0.0.4

10.0.0.4

10.0.0.5



```
control ingress {

    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        apply(address_arp_packet);
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        apply(address_ip_packet);
    }
    if ((ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) or (ether
      apply(vpc_customer);
    }
    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        if (ingress_metadata.customer > 0){
          apply(arp_reply);
        }
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        if (ingress_metadata.customer > 0){
          apply(encapsulate_vpc);
          apply(vpc_sw_id);
          apply(vpc_dst);
        }
    }
```

L2 – Ethernet (using P4 VPC)

Com

oute

10.0

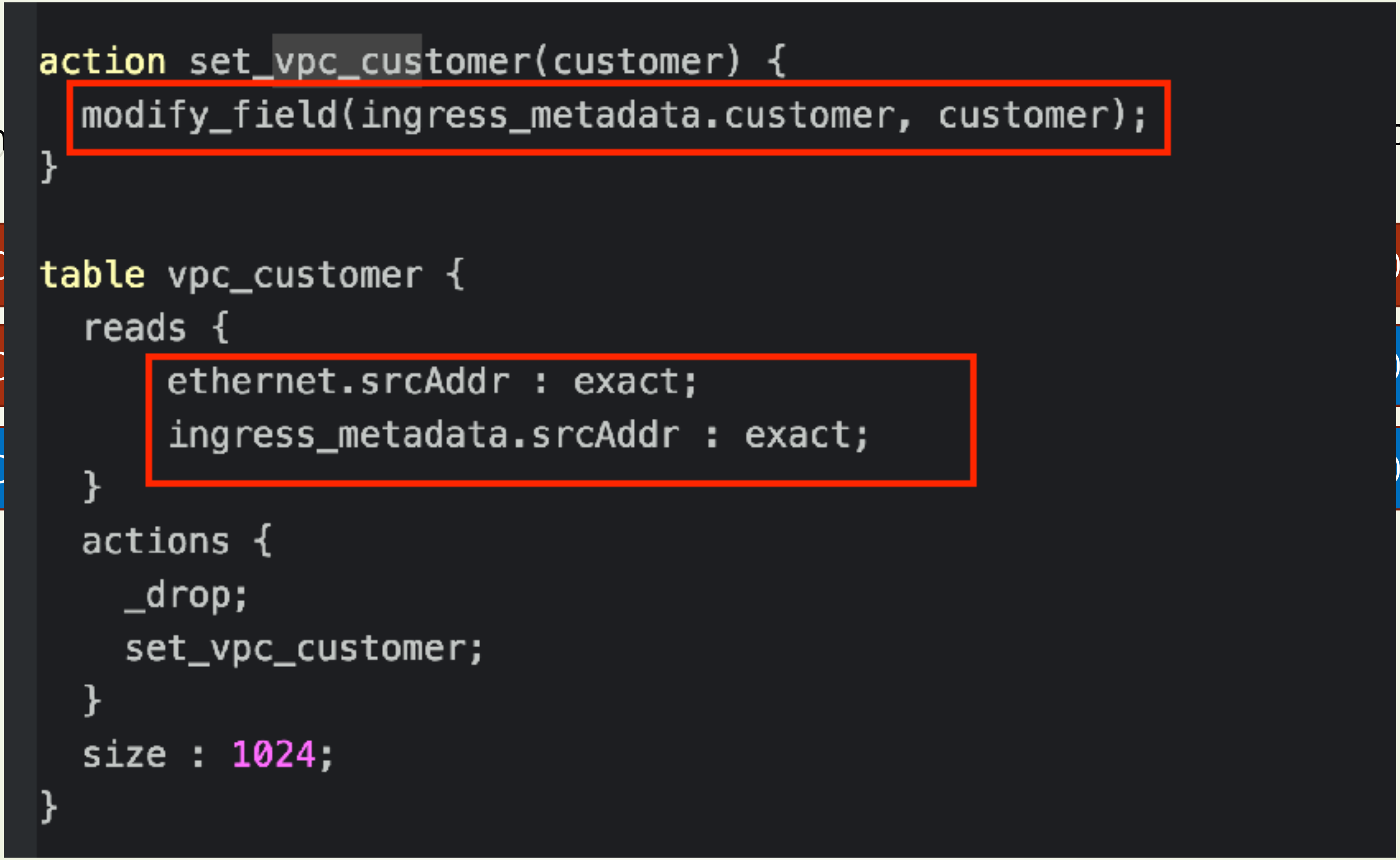
.0.4

10.0

.0.4

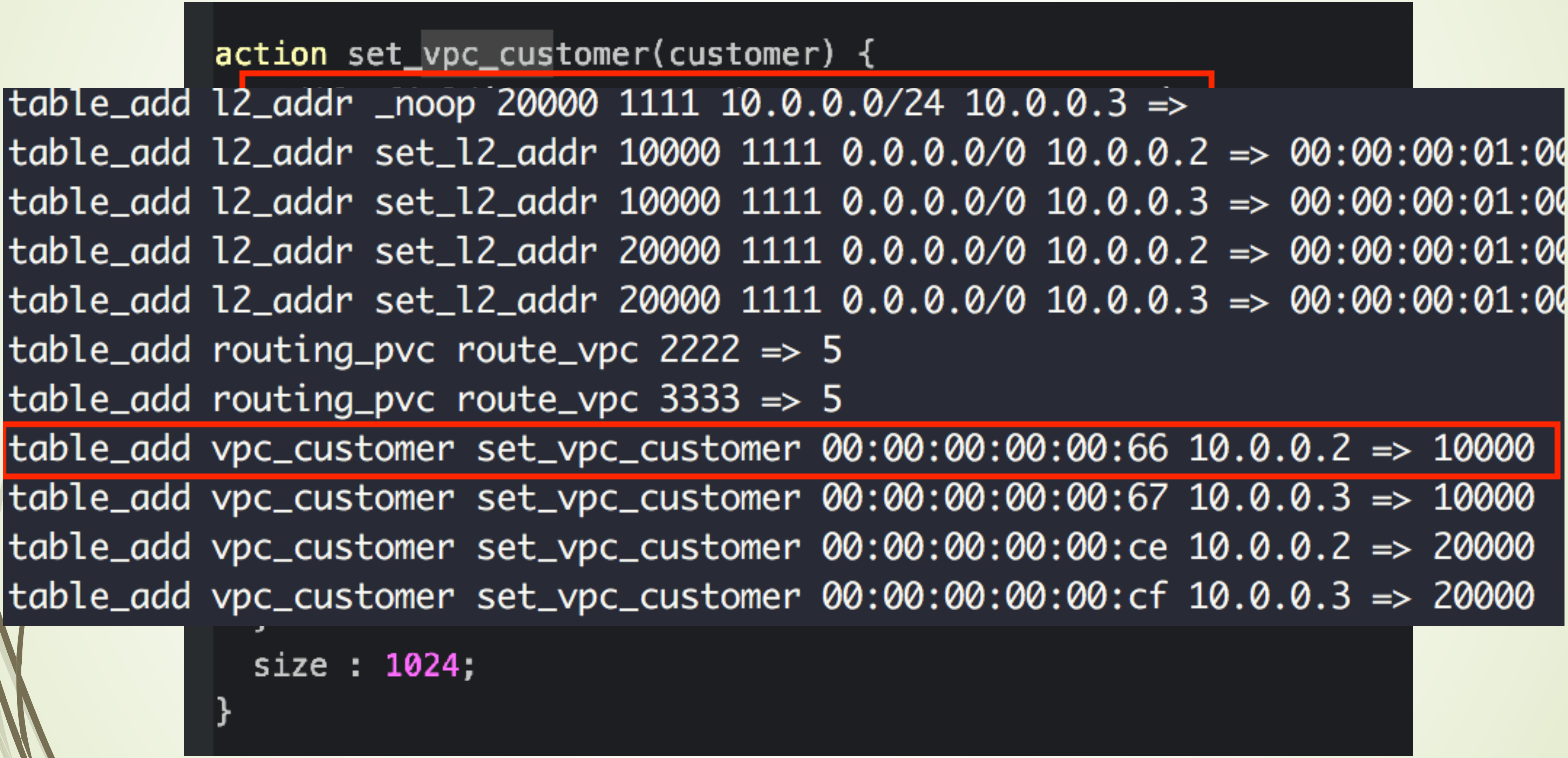
10.0

.0.5



```
action set_vpc_customer(customer) {
  modify_field(ingress_metadata.customer, customer);
}

table vpc_customer {
  reads {
      ethernet.srcAddr : exact;
      ingress_metadata.srcAddr : exact;
  }
  actions {
    _drop;
    set_vpc_customer;
  }
  size : 1024;
}
```

L2 – Ethernet (using P4 VPC)



```
action set_vpc_customer(customer) {
```

```
table_add l2_addr _noop 20000 1111 10.0.0.0/24 10.0.0.3 =>
table_add l2_addr set_l2_addr 10000 1111 0.0.0.0/0 10.0.0.2 => 00:00:00:01:00
table_add l2_addr set_l2_addr 10000 1111 0.0.0.0/0 10.0.0.3 => 00:00:00:01:00
table_add l2_addr set_l2_addr 20000 1111 0.0.0.0/0 10.0.0.2 => 00:00:00:01:00
table_add l2_addr set_l2_addr 20000 1111 0.0.0.0/0 10.0.0.3 => 00:00:00:01:00
table_add routing_pvc route_vpc 2222 => 5
table_add routing_pvc route_vpc 3333 => 5
table_add vpc_customer set_vpc_customer 00:00:00:00:00:66 10.0.0.2 => 10000
table_add vpc_customer set_vpc_customer 00:00:00:00:00:67 10.0.0.3 => 10000
table_add vpc_customer set_vpc_customer 00:00:00:00:00:ce 10.0.0.2 => 20000
table_add vpc_customer set_vpc_customer 00:00:00:00:00:cf 10.0.0.3 => 20000
```

```
  size : 1024;
}
```

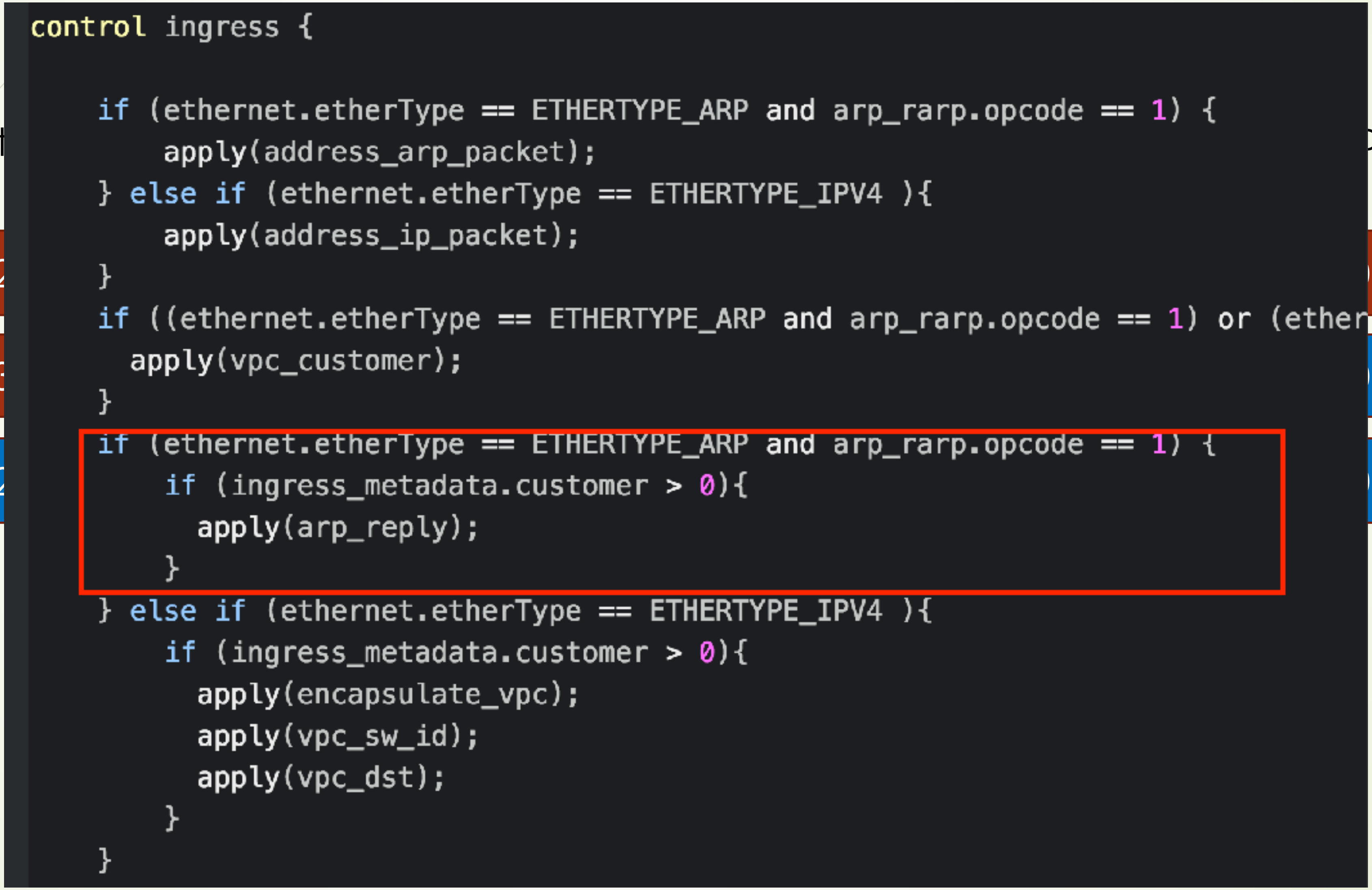
L2 – Ethernet (using P4 VPC)

Comput

10.0.0.2

10.0.0.3

10.0.0.2



```
control ingress {

    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        apply(address_arp_packet);
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        apply(address_ip_packet);
    }
    if ((ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) or (ether
      apply(vpc_customer);
    }
    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        if (ingress_metadata.customer > 0){
          apply(arp_reply);
        }
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        if (ingress_metadata.customer > 0){
          apply(encapsulate_vpc);
          apply(vpc_sw_id);
          apply(vpc_dst);
        }
    }
```

ute

.0.4

.0.4

.0.5

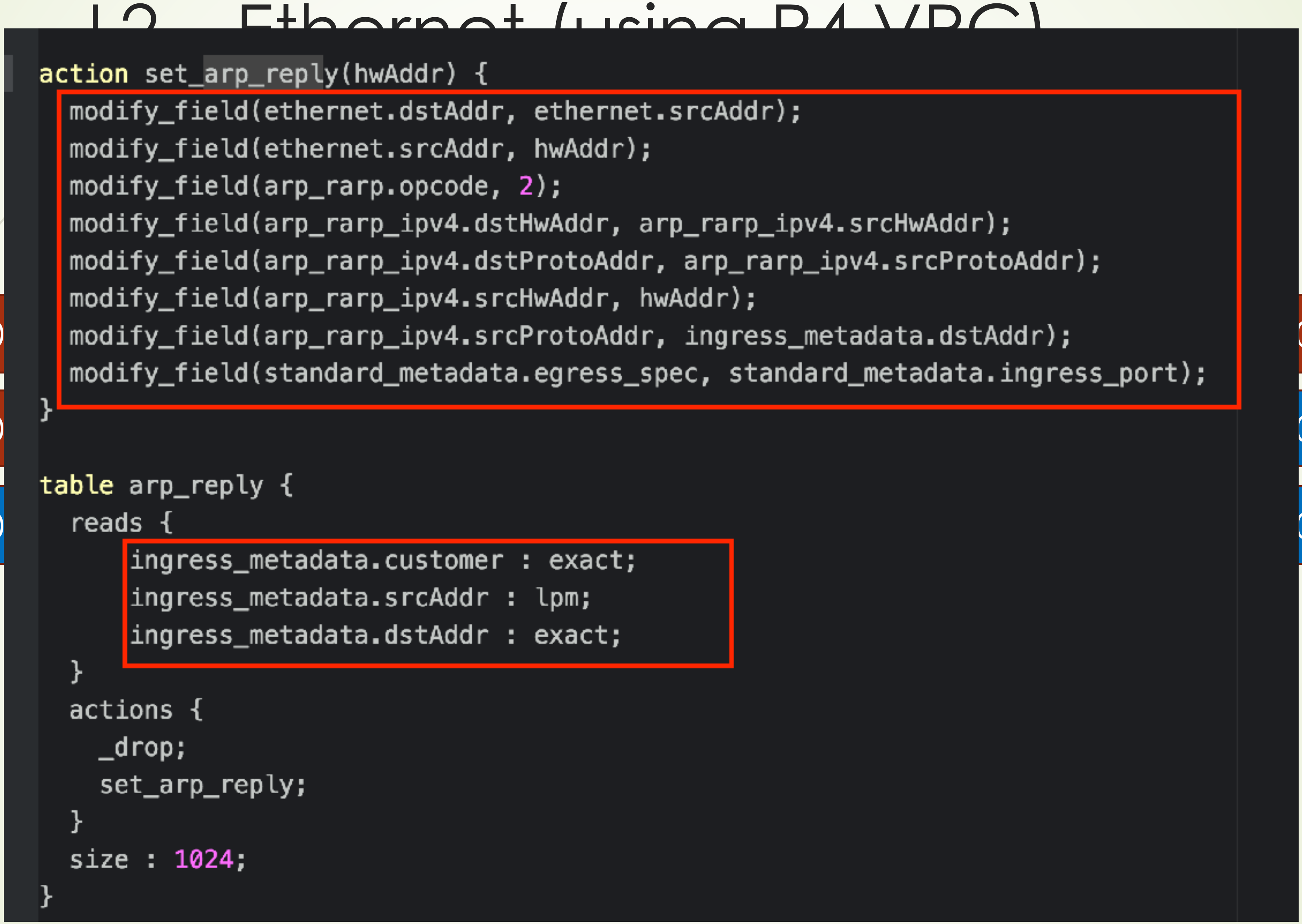
L2 – Ethernet (using P4 VPC)

Com

10.0

10.0

10.0



```
action set_arp_reply(hwAddr) {
  modify_field(ethernet.dstAddr, ethernet.srcAddr);
  modify_field(ethernet.srcAddr, hwAddr);
  modify_field(arp_rarp.opcode, 2);
  modify_field(arp_rarp_ipv4.dstHwAddr, arp_rarp_ipv4.srcHwAddr);
  modify_field(arp_rarp_ipv4.dstProtoAddr, arp_rarp_ipv4.srcProtoAddr);
  modify_field(arp_rarp_ipv4.srcHwAddr, hwAddr);
  modify_field(arp_rarp_ipv4.srcProtoAddr, ingress_metadata.dstAddr);
  modify_field(standard_metadata.egress_spec, standard_metadata.ingress_port);
}

table arp_reply {
  reads {
      ingress_metadata.customer : exact;
      ingress_metadata.srcAddr : lpm;
      ingress_metadata.dstAddr : exact;
  }
  actions {
    _drop;
    set_arp_reply;
  }
  size : 1024;
}
```

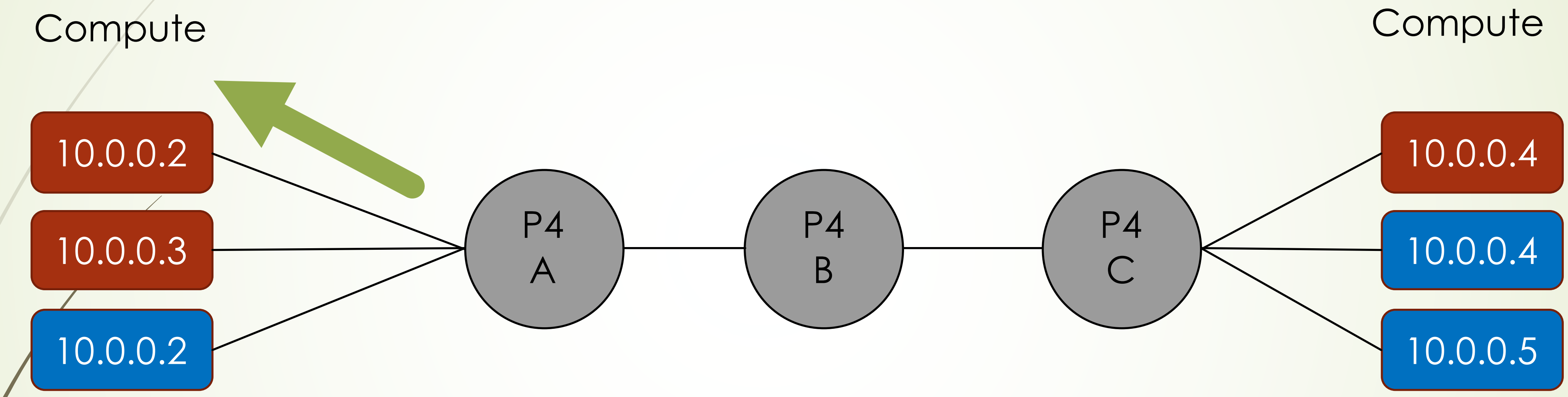
pute

0.0.4

0.0.4

0.0.5

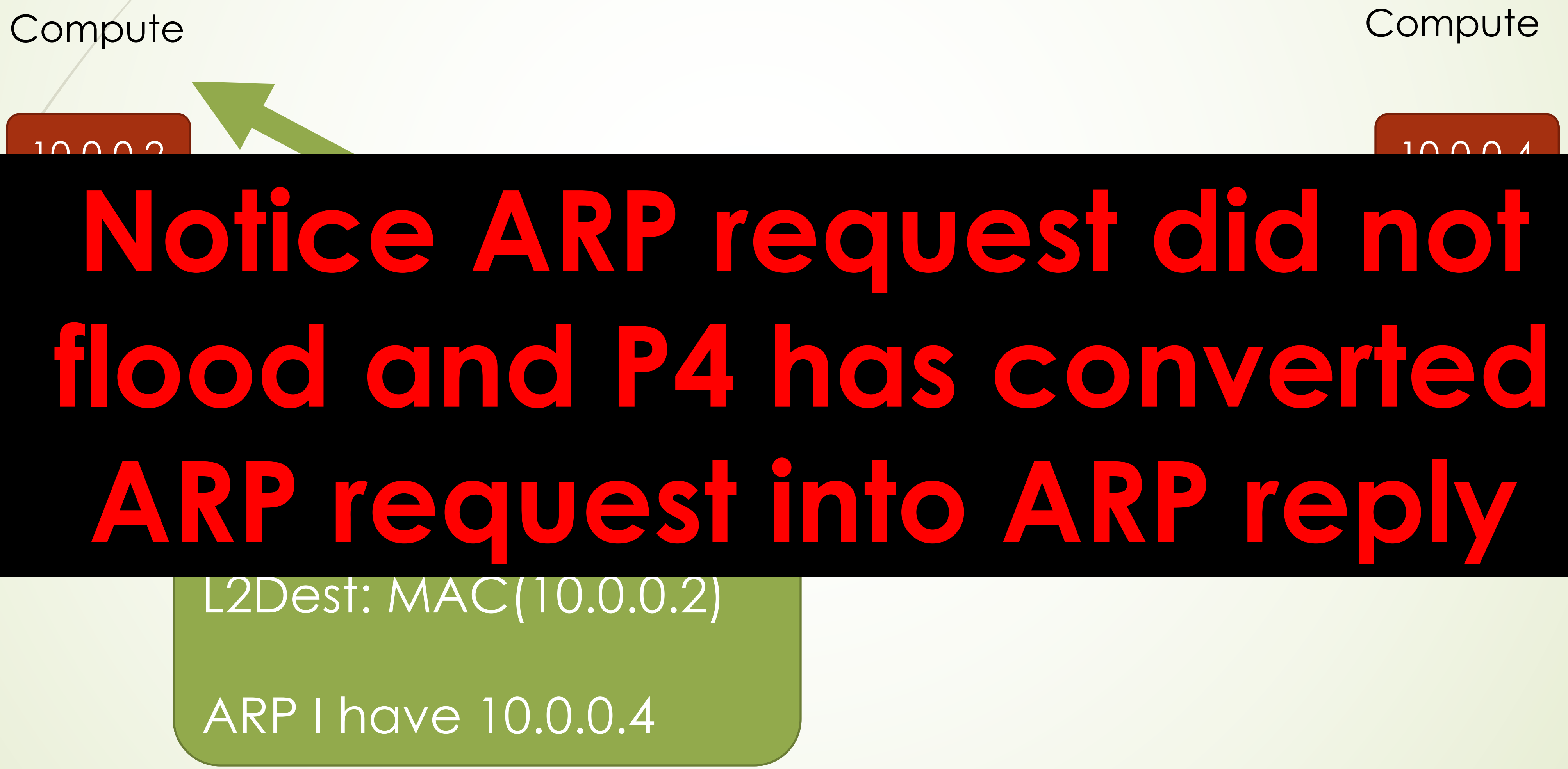
L2 – Ethernet (using P4 VPC)



L2Src: MAC(10.0.0.4)
L2Dest: MAC(10.0.0.2)

ARP I have 10.0.0.4

L2 – Ethernet (using P4 VPC)

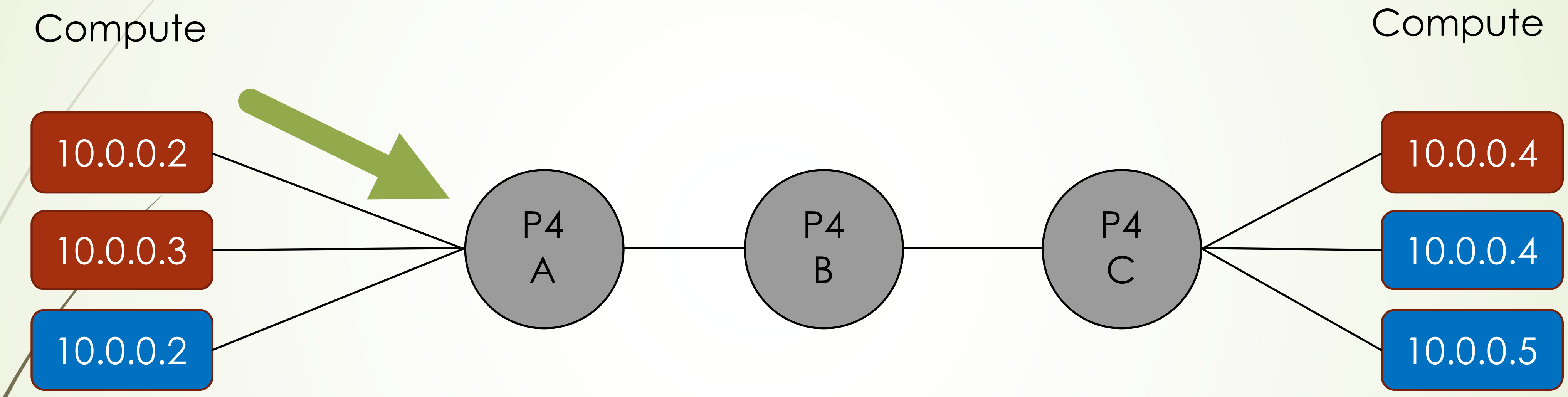


L2Dest: MAC(10.0.0.2)

ARP I have 10.0.0.4

Notice ARP request did not flood and P4 has converted ARP request into ARP reply

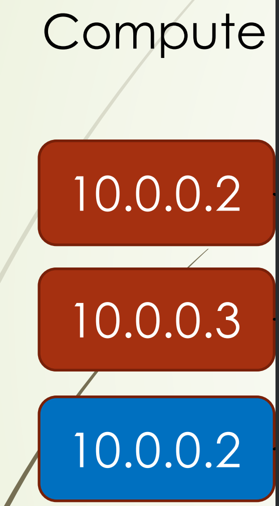
L2 – Ethernet (using P4 VPC)

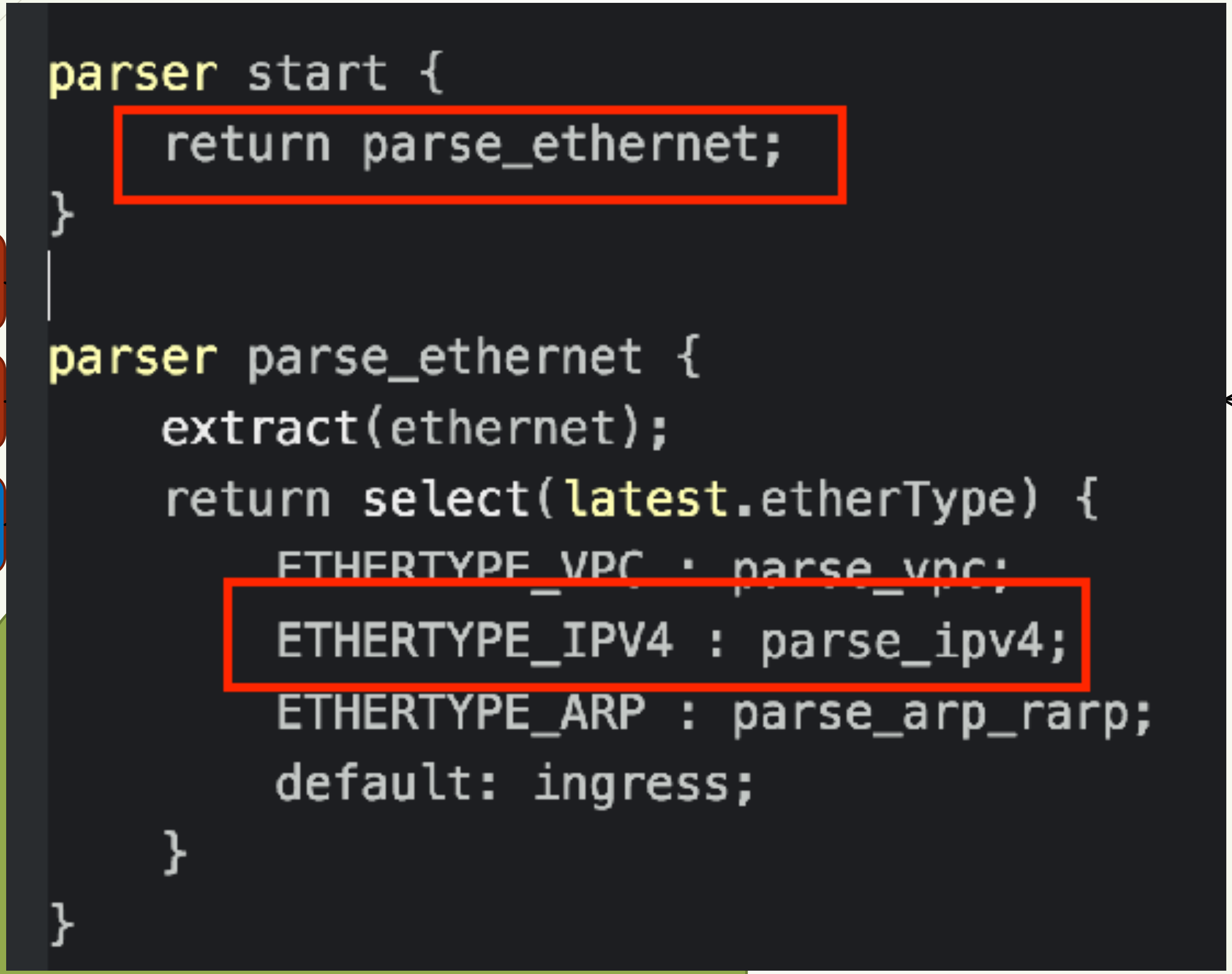


L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.4)

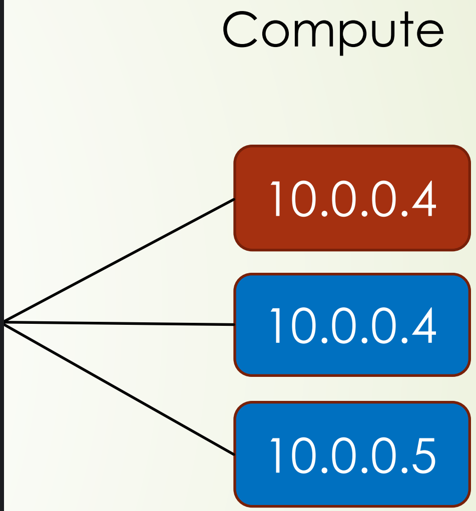
ICMP answer to10.0.2

L2 – Ethernet (using P4 VPC)





```
parser start {
    return parse_ethernet;
}

parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        ETHERTYPE_VPC : parse_vpc;
        ETHERTYPE_IPV4 : parse_ipv4;
        ETHERTYPE_ARP : parse_arp_rarp;
        default: ingress;
    }
}
```



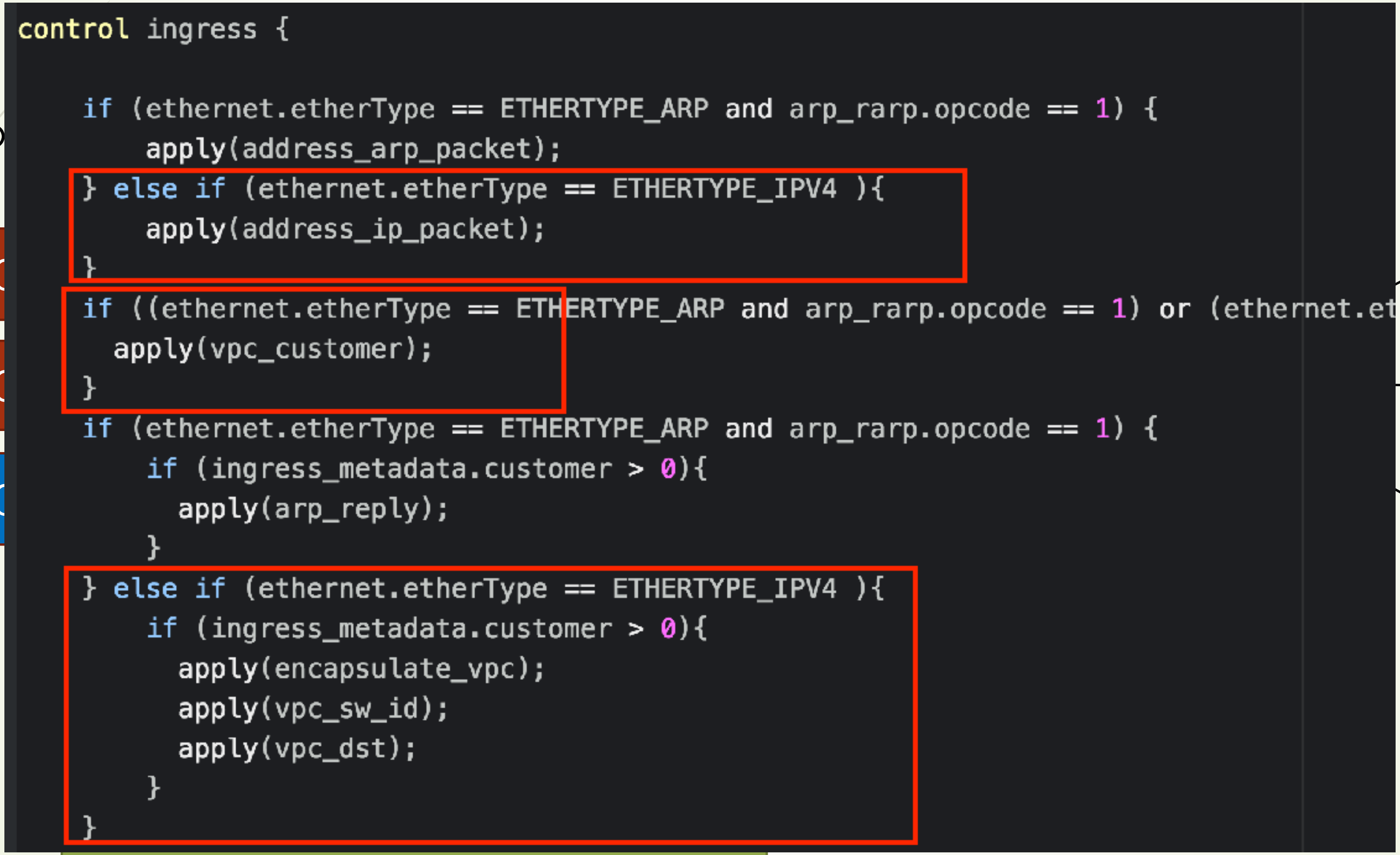
L2 – Ethernet (using P4 VPC)

Comp

10.0.(

10.0.(

10.0.(



```
control ingress {

    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        apply(address_arp_packet);
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        apply(address_ip_packet);
    }
    if ((ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) or (ethernet.et
      apply(vpc_customer);
    }
    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        if (ingress_metadata.customer > 0){
          apply(arp_reply);
        }
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        if (ingress_metadata.customer > 0){
          apply(encapsulate_vpc);
          apply(vpc_sw_id);
          apply(vpc_dst);
        }
    }
```

Compute

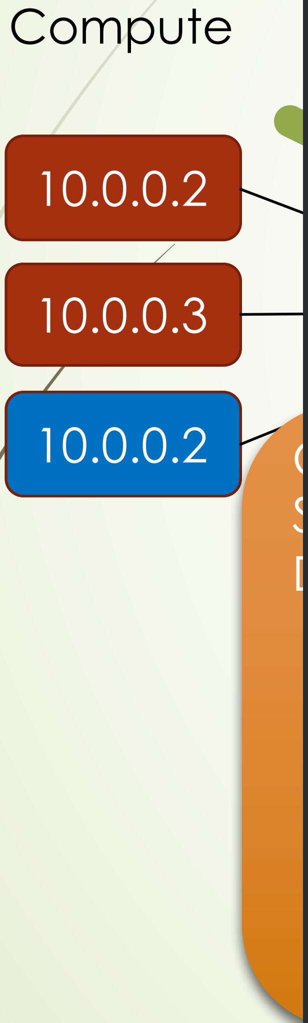
10.0.0.4

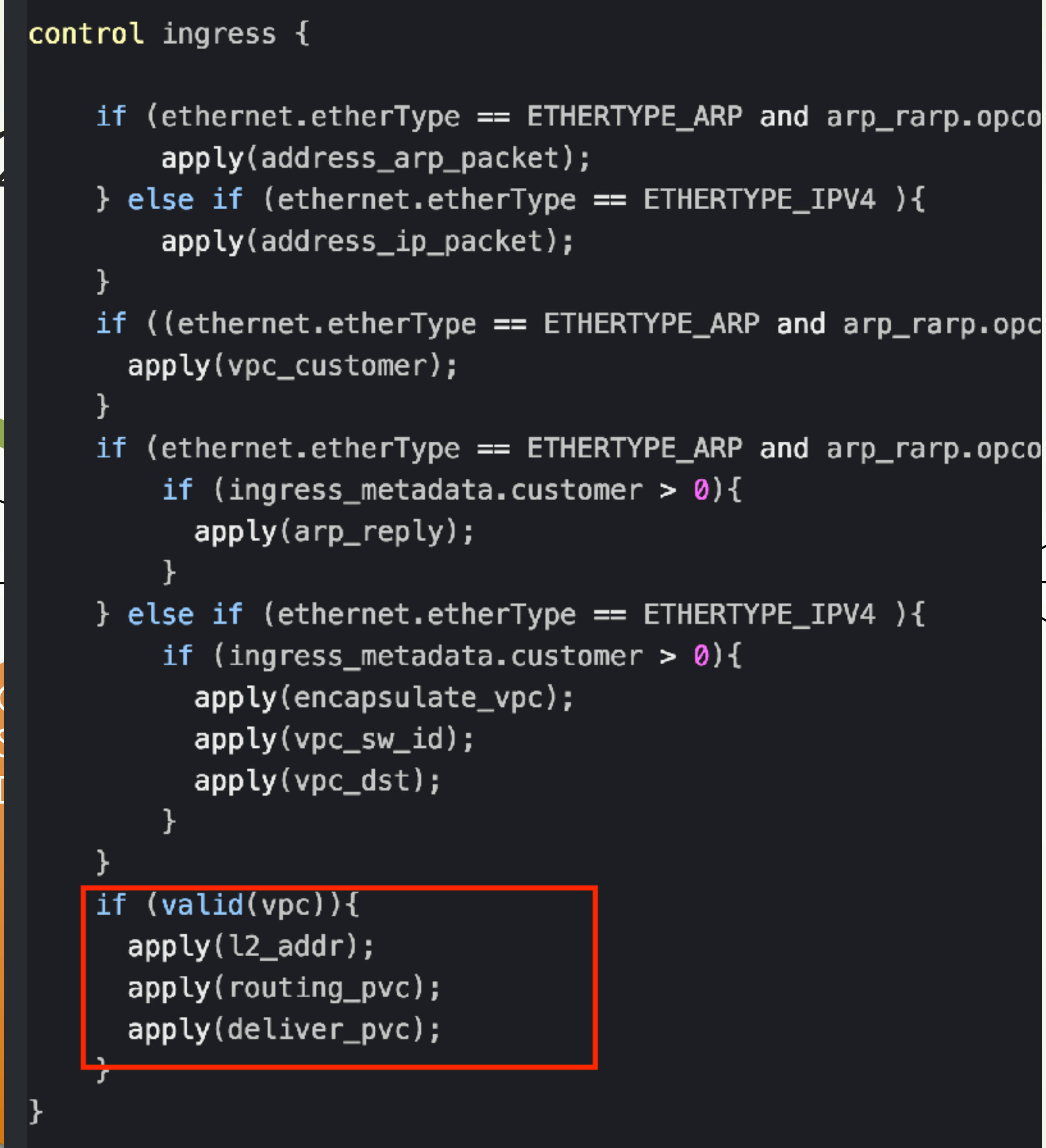
10.0.0.4

10.0.0.5

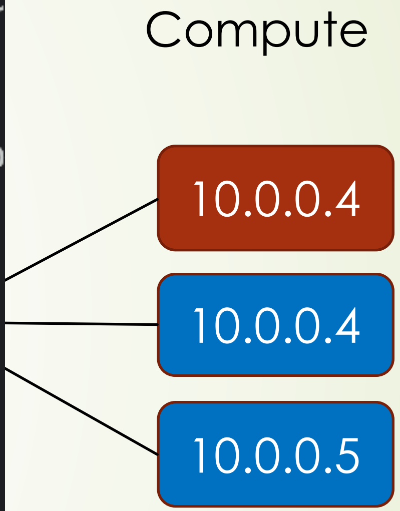
ICMP answer to10.0.2

L2

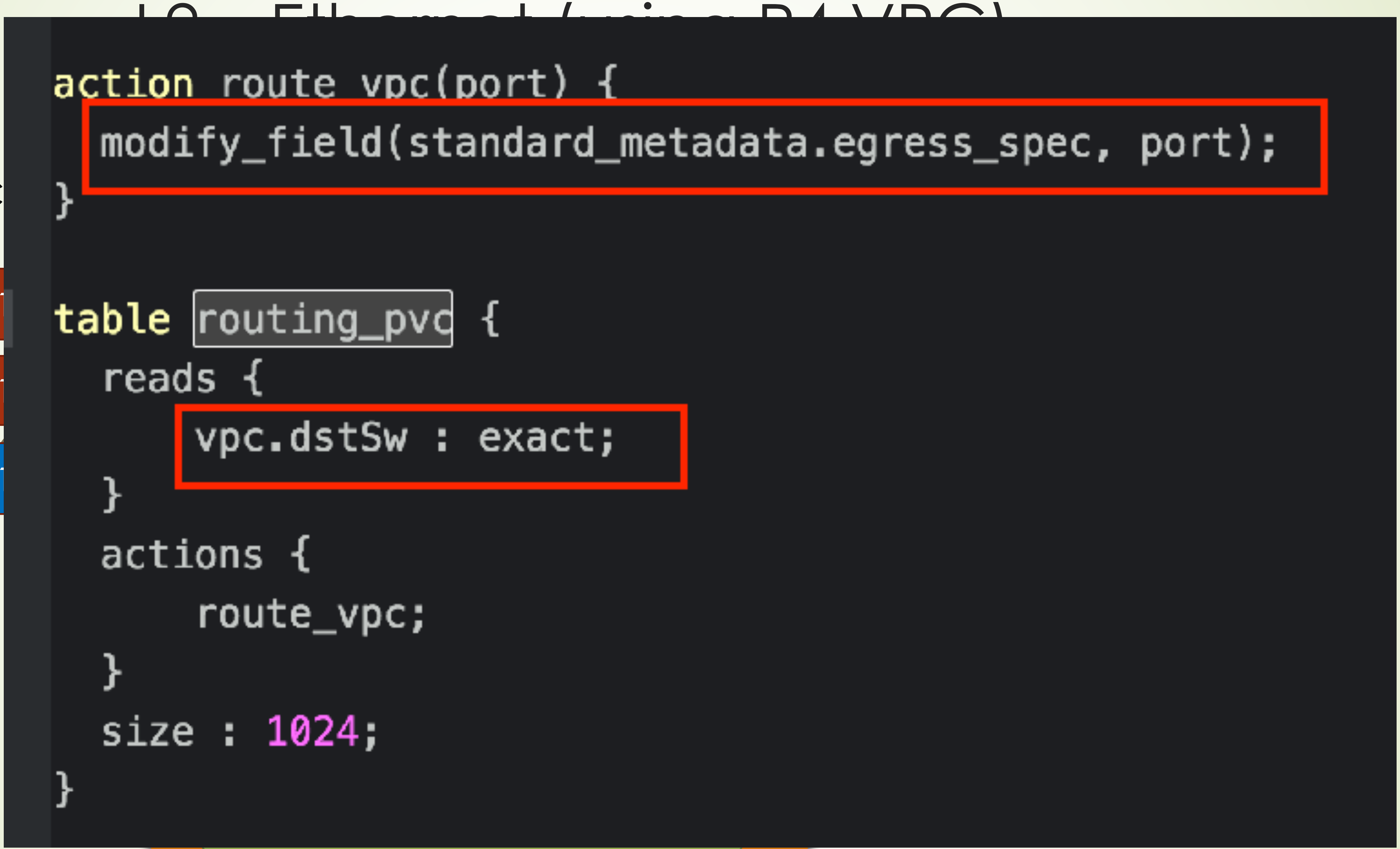




```
control ingress {

    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opco
        apply(address_arp_packet);
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        apply(address_ip_packet);
    }
    if ((ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opc
      apply(vpc_customer);
    }
    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opco
        if (ingress_metadata.customer > 0){
          apply(arp_reply);
        }
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        if (ingress_metadata.customer > 0){
          apply(encapsulate_vpc);
          apply(vpc_sw_id);
          apply(vpc_dst);
        }
    }
    if (valid(vpc)){
      apply(l2_addr);
      apply(routing_pvc);
      apply(deliver_pvc);
    }
}
```

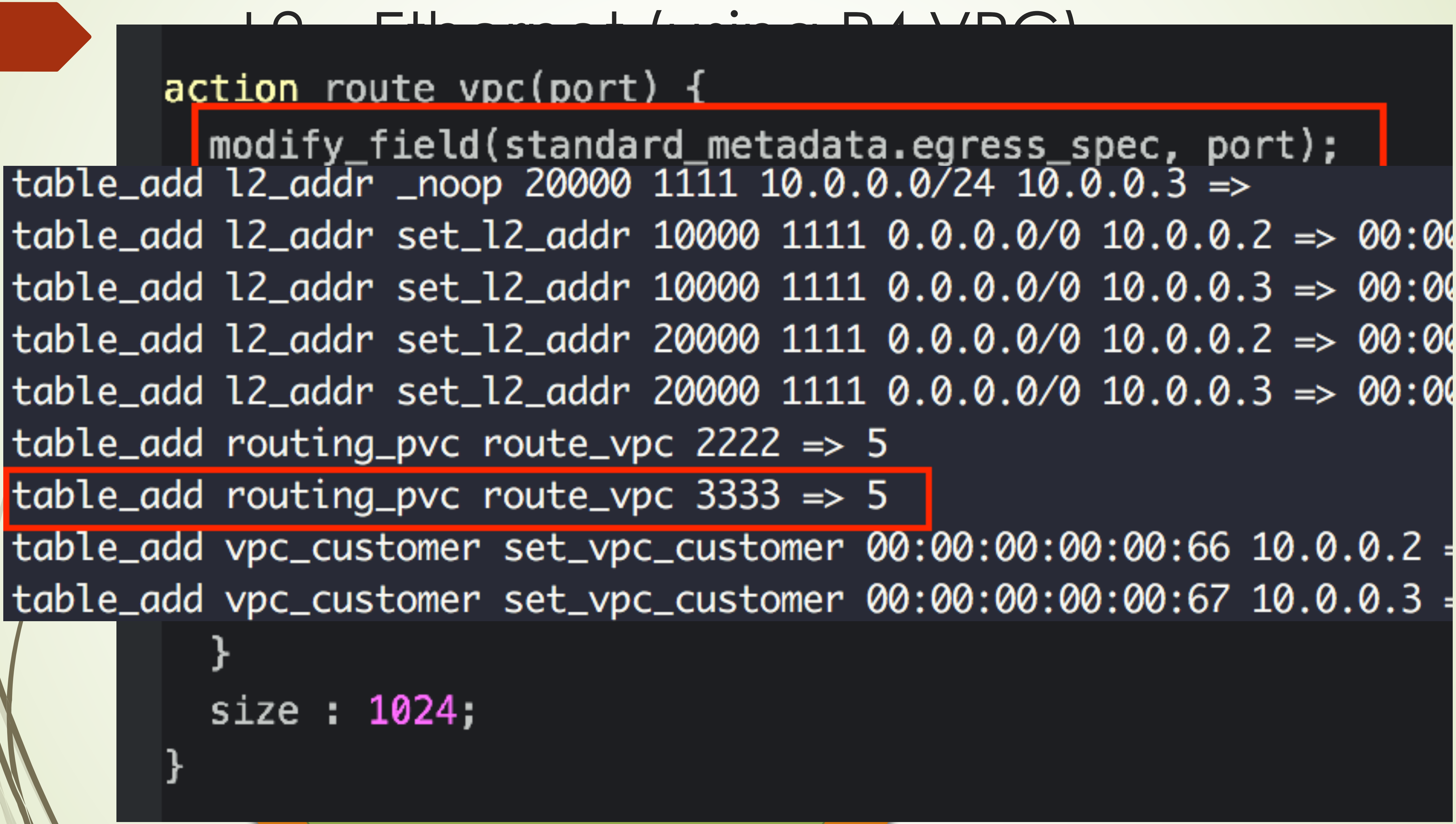


L2 – Ethernet (using P4 VPC)



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

table routing_pvc {
  reads {
      vpc.dstSw : exact;
  }
  actions {
      route_vpc;
  }
  size : 1024;
}
```

L2 – Ethernet (using P4 VPC)

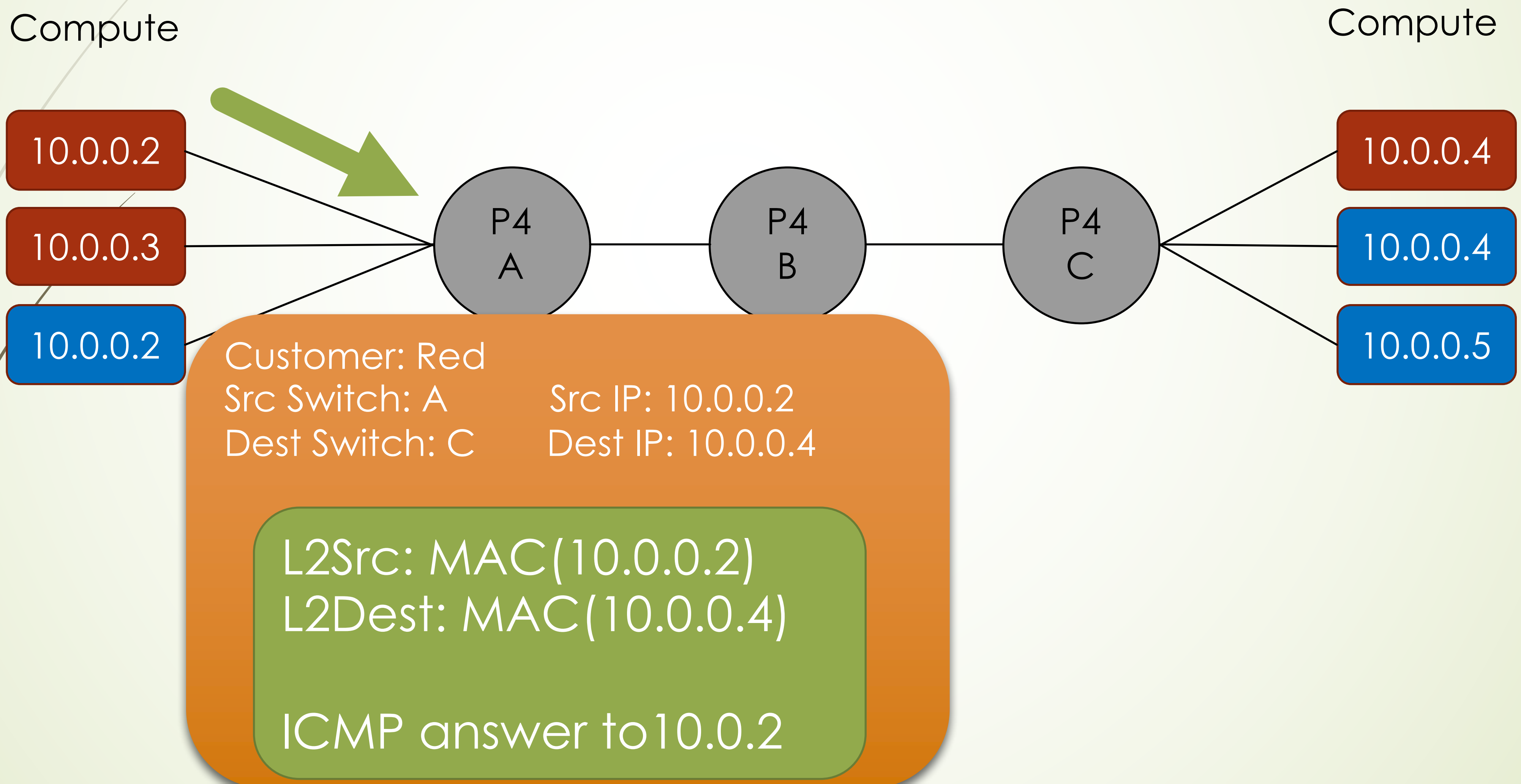


```
action route_vpc(port) {
    modify_field(standard_metadata.egress_spec, port);
```

```
table_add l2_addr _noop 20000 1111 10.0.0.0/24 10.0.0.3 =>
table_add l2_addr set_l2_addr 10000 1111 0.0.0.0/0 10.0.0.2 => 00:00
table_add l2_addr set_l2_addr 10000 1111 0.0.0.0/0 10.0.0.3 => 00:00
table_add l2_addr set_l2_addr 20000 1111 0.0.0.0/0 10.0.0.2 => 00:00
table_add l2_addr set_l2_addr 20000 1111 0.0.0.0/0 10.0.0.3 => 00:00
table_add routing_pvc route_vpc 2222 => 5
table_add routing_pvc route_vpc 3333 => 5
table_add vpc_customer set_vpc_customer 00:00:00:00:00:66 10.0.0.2 =
table_add vpc_customer set_vpc_customer 00:00:00:00:00:67 10.0.0.3 =
```

```
    }
    size : 1024;
}
```

L2 – Ethernet (using P4 VPC)



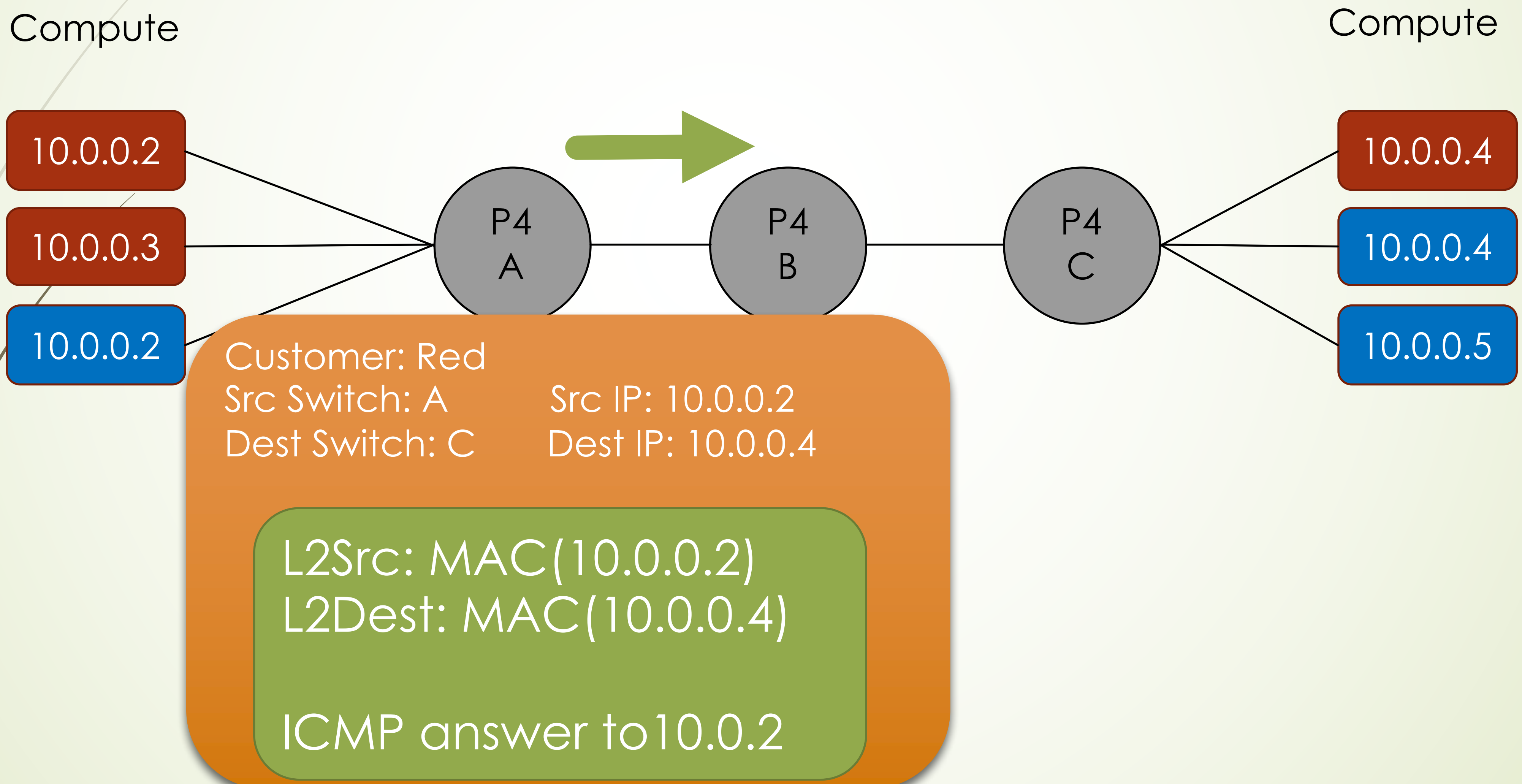
Customer: Red
Src Switch: A
Dest Switch: C

Src IP: 10.0.0.2
Dest IP: 10.0.0.4

L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.4)

ICMP answer to10.0.2

L2 – Ethernet (using P4 VPC)



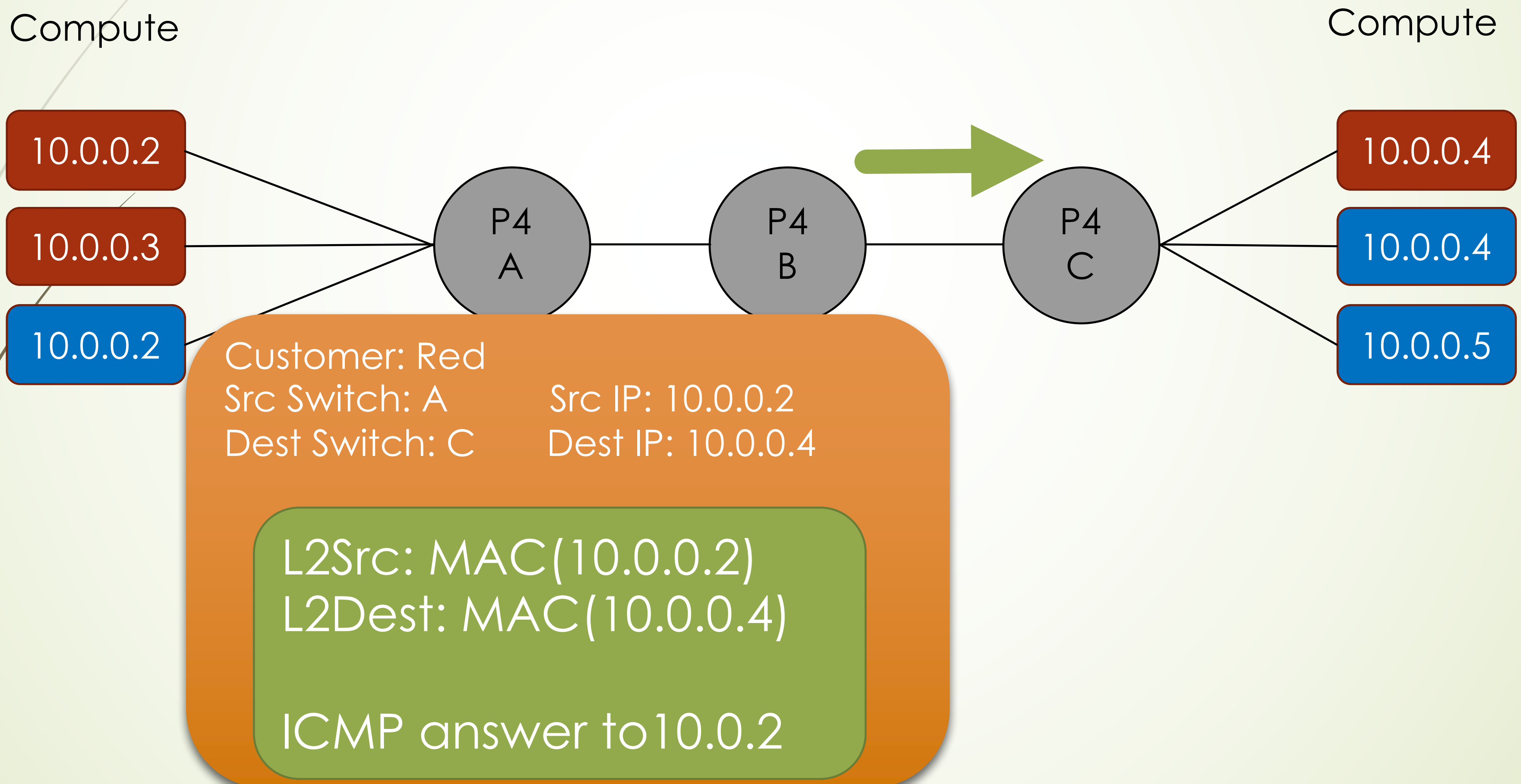
Customer: Red
Src Switch: A
Dest Switch: C

Src IP: 10.0.0.2
Dest IP: 10.0.0.4

L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.4)

ICMP answer to10.0.2

L2 – Ethernet (using P4 VPC)



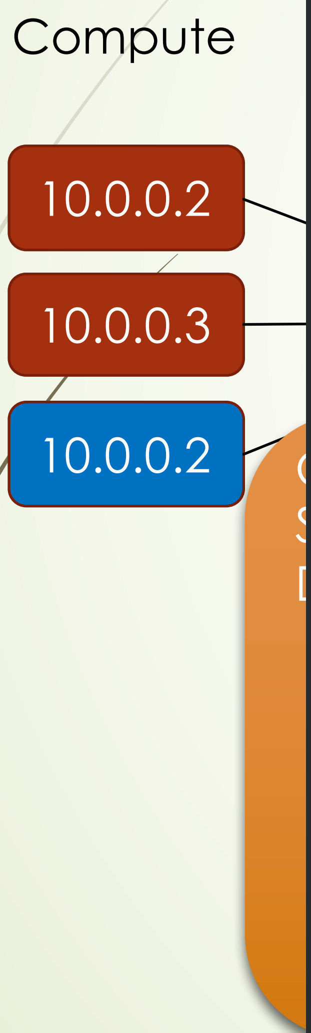
Customer: Red
Src Switch: A
Dest Switch: C

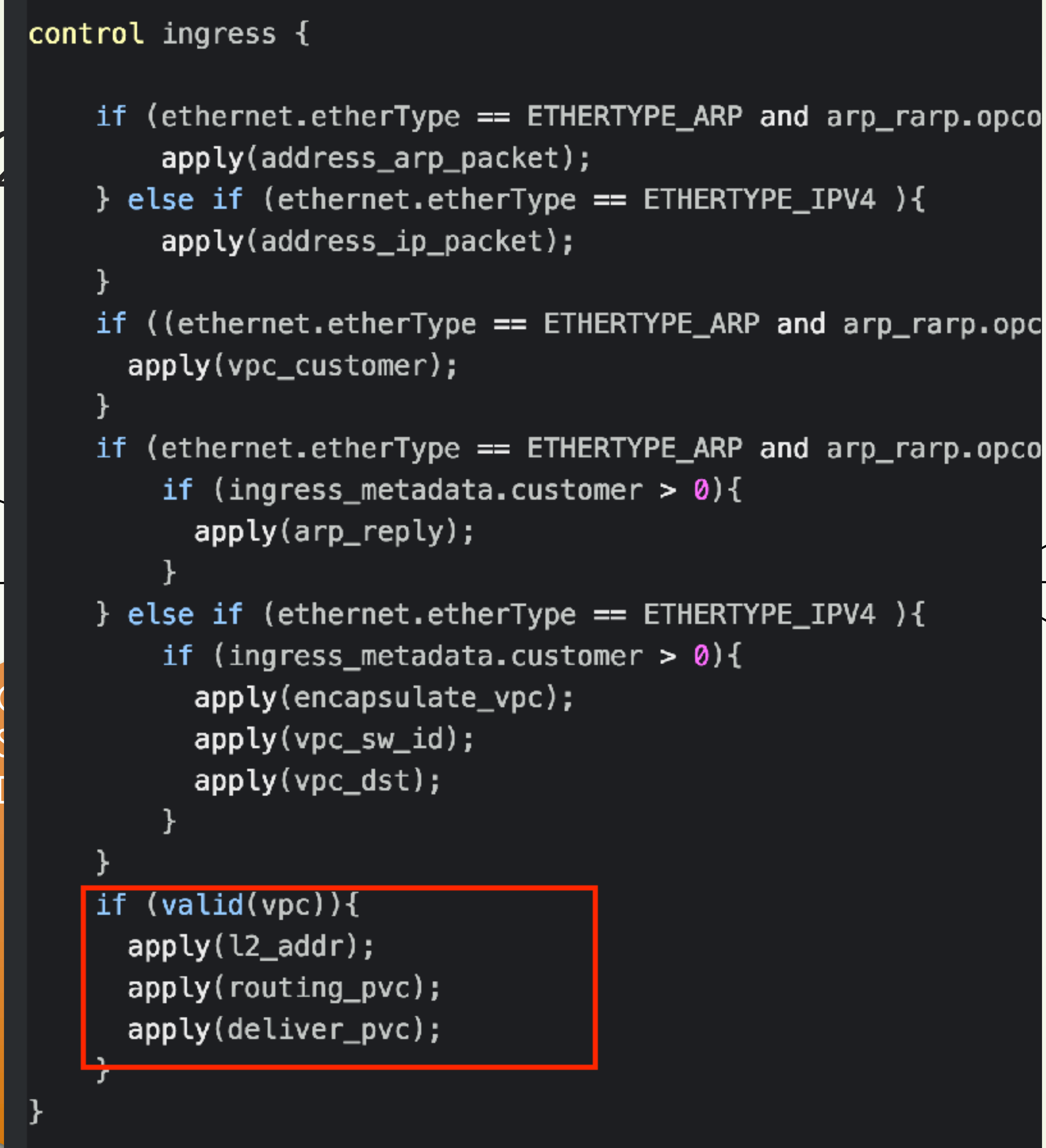
Src IP: 10.0.0.2
Dest IP: 10.0.0.4

L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.4)

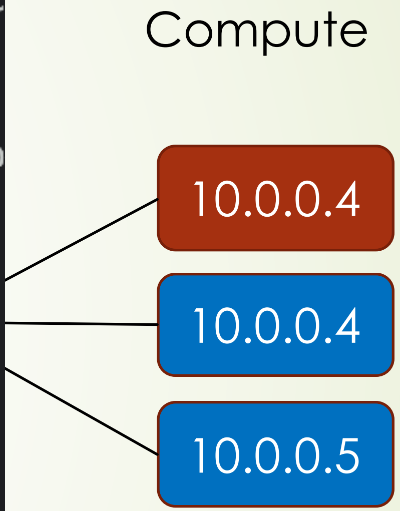
ICMP answer to10.0.2

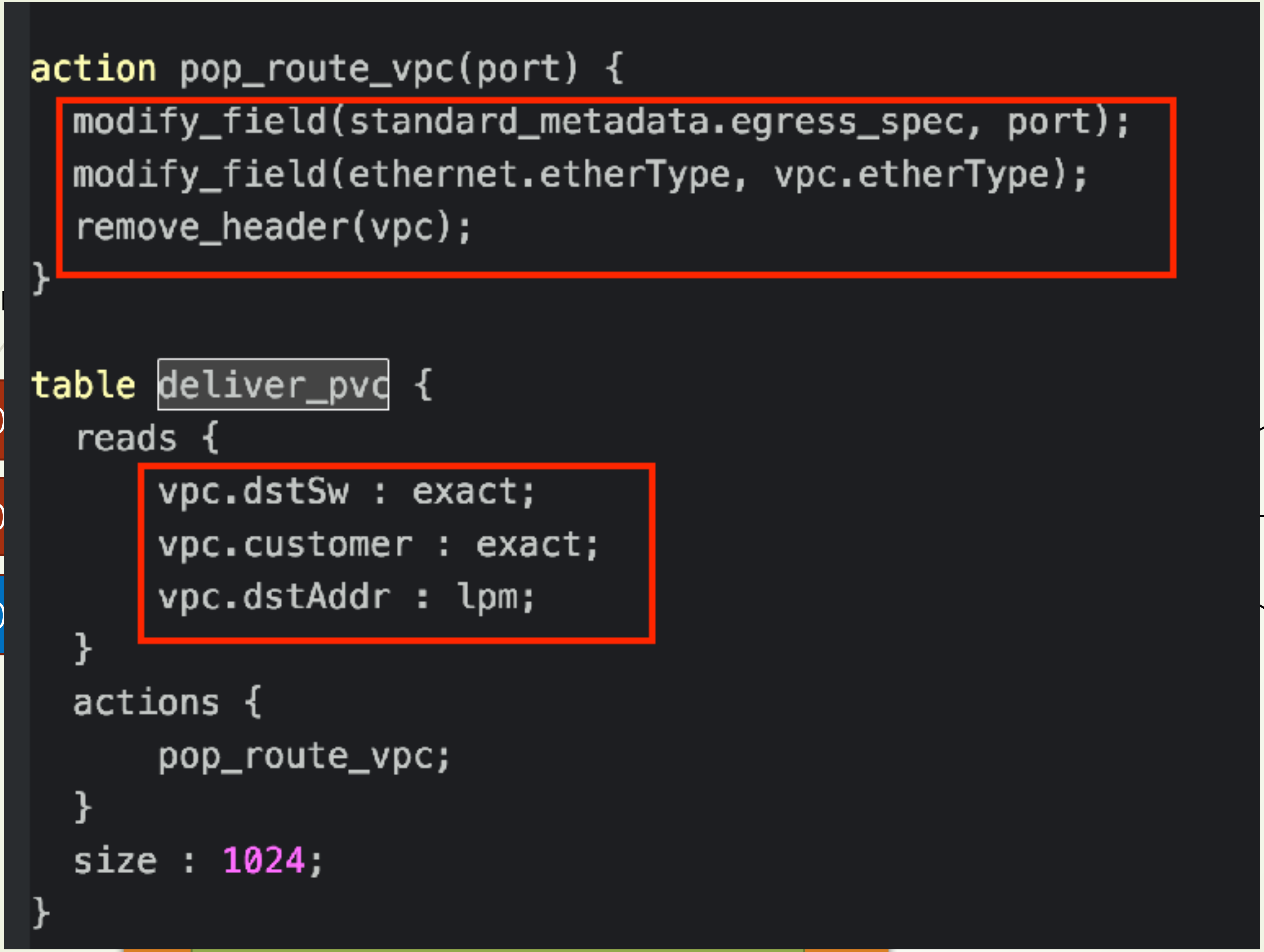
L2





```
control ingress {

    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opco
        apply(address_arp_packet);
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        apply(address_ip_packet);
    }
    if ((ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opc
      apply(vpc_customer);
    }
    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opco
        if (ingress_metadata.customer > 0){
          apply(arp_reply);
        }
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        if (ingress_metadata.customer > 0){
          apply(encapsulate_vpc);
          apply(vpc_sw_id);
          apply(vpc_dst);
        }
    }
    if (valid(vpc)){
      apply(l2_addr);
      apply(routing_pvc);
      apply(deliver_pvc);
    }
}
```





```
action pop_route_vpc(port) {
  modify_field(standard_metadata.egress_spec, port);
  modify_field(ethernet.etherType, vpc.etherType);
  remove_header(vpc);
}

table deliver_pvc {
  reads {
      vpc.dstSw : exact;
      vpc.customer : exact;
      vpc.dstAddr : lpm;
  }
  actions {
      pop_route_vpc;
  }
  size : 1024;
}
```

Co

10

10

10

Compute

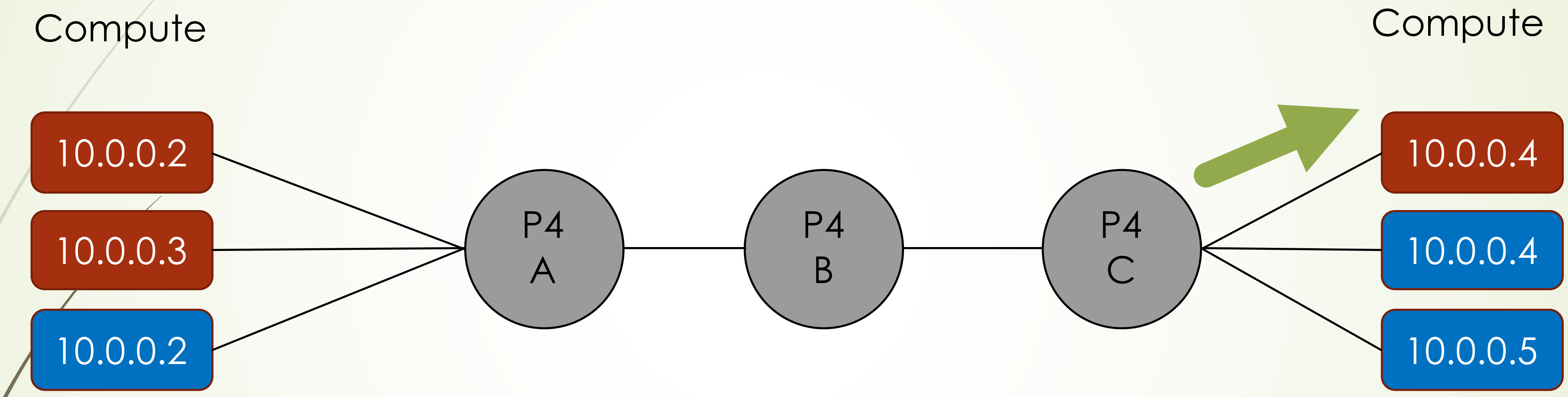
10.0.0.4

10.0.0.4

10.0.0.5

ICMP answer to10.0.2

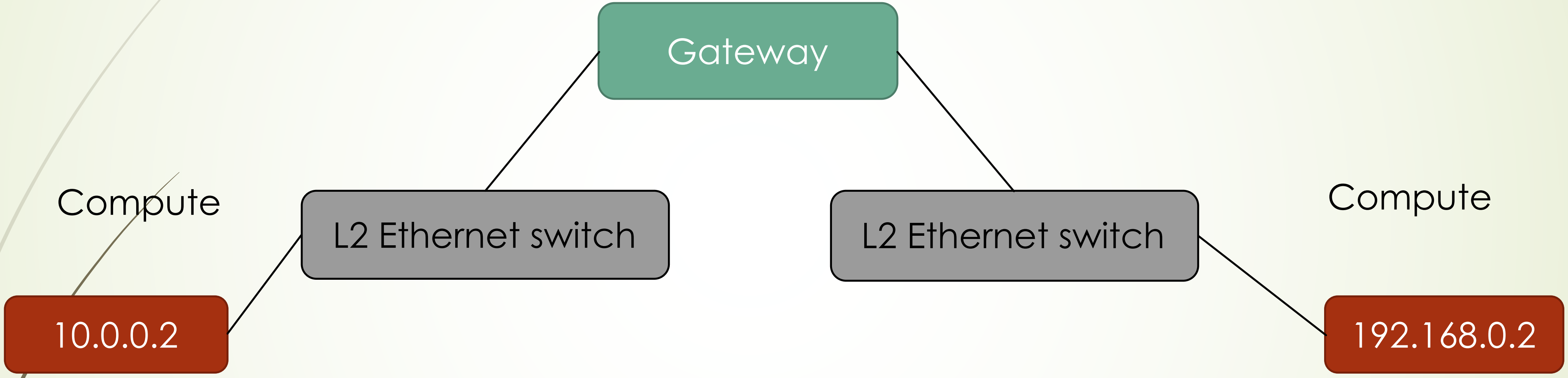
L2 – Ethernet (using P4 VPC)



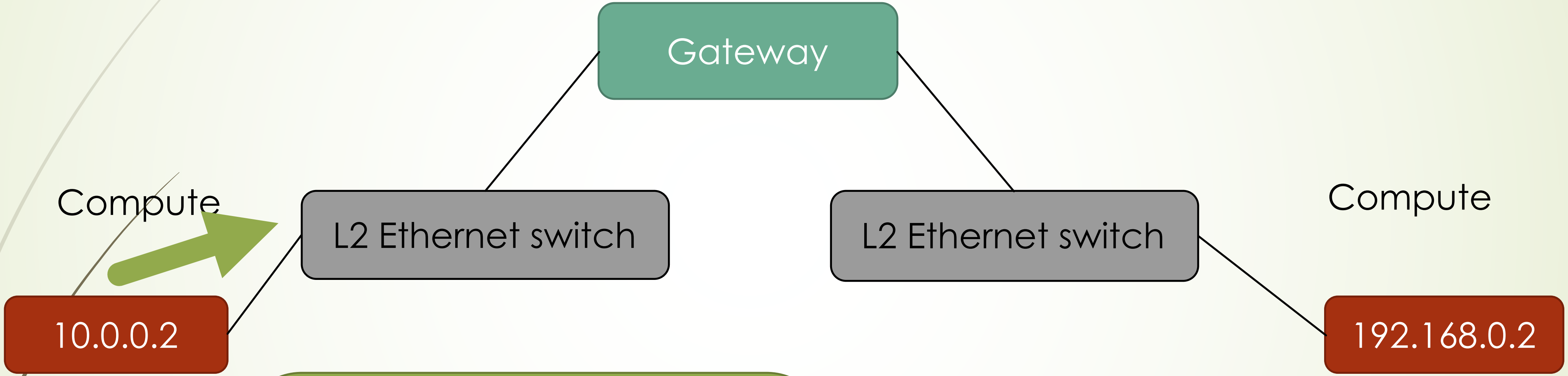
L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.4)

ICMP answer to10.0.2

L3 – IP Routing(standard)



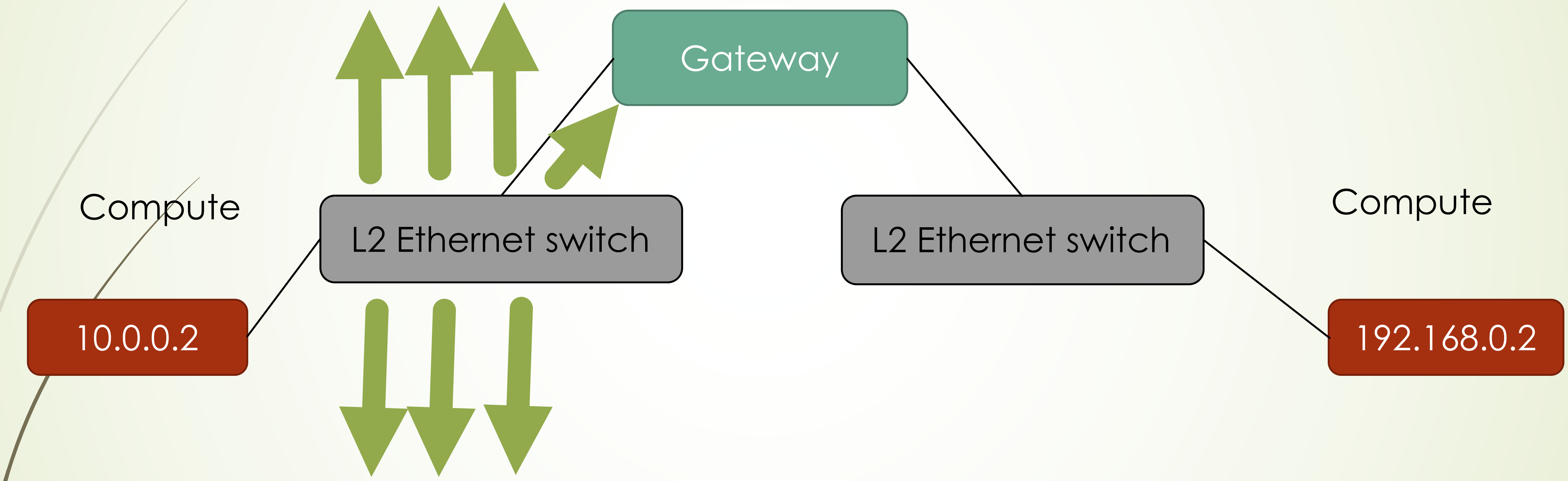
L3 – IP Routing(standard)



L2Src: MAC(10.0.0.2)
L2Dest: FF:FF:FF:FF:FF:FF

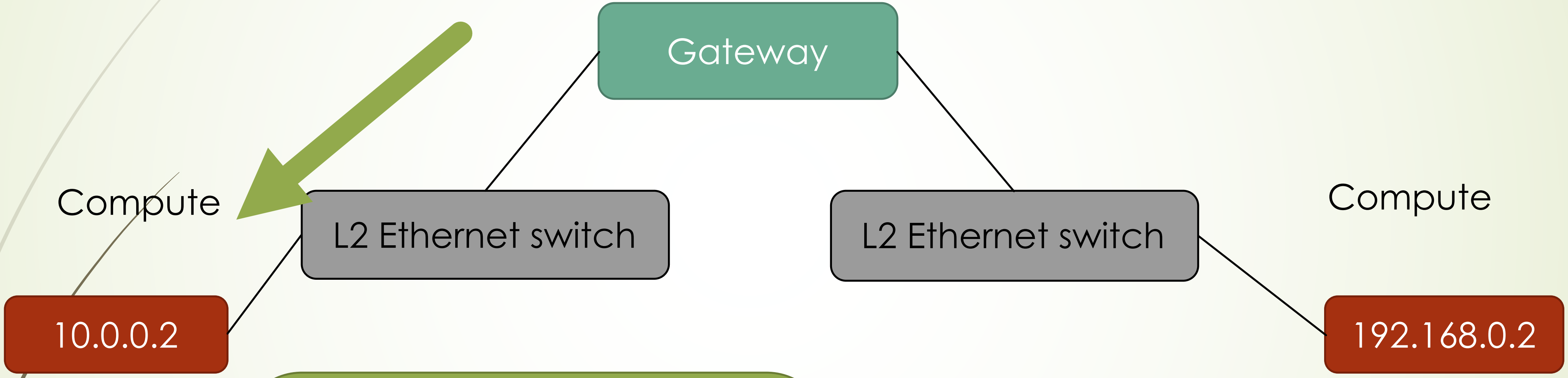
ARP who has 10.0.0.1

L3 – IP Routing(standard)



Packet is flooded

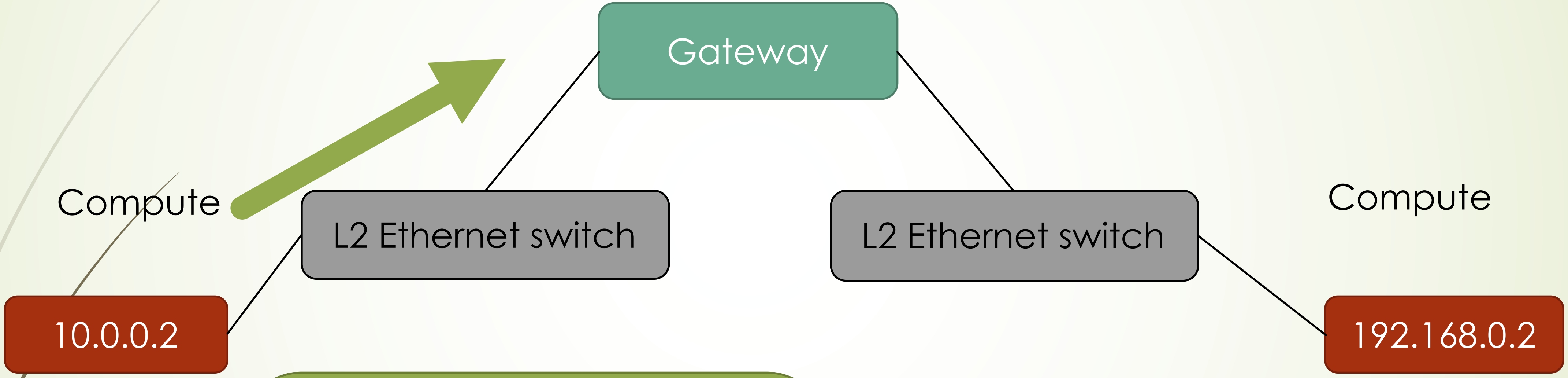
L3 – IP Routing(standard)



L2Src: MAC(10.0.0.1)
L2Dest: MAC(10.0.0.2)

ARP I have 10.0.0.1

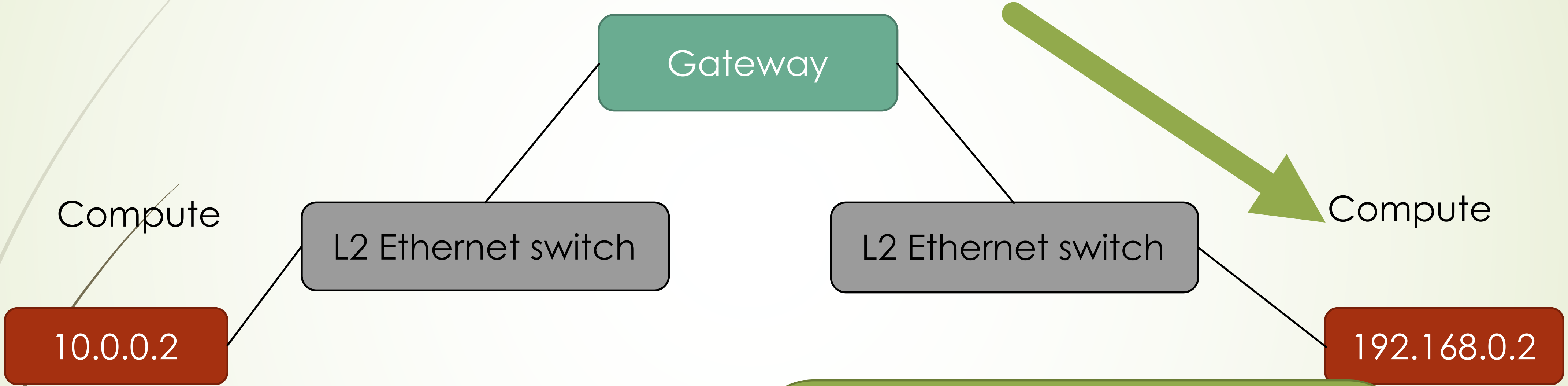
L3 – IP Routing(standard)



L2Src: MAC (10.0.0.2)
L2Dest: MAC (10.0.0.1)

ICMP to 192.168.0.2
answer to 10.0.0.2

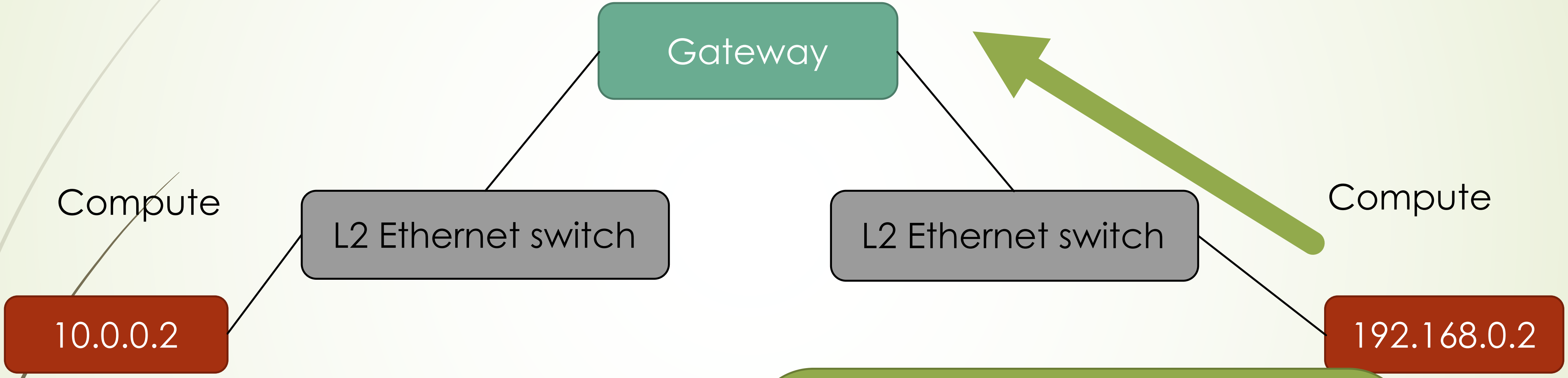
L3 – IP Routing(standard)



L2Src: MAC (192.168.0.1)
L2Dest: MAC (192.168.0.2)

ICMP to 192.168.0.2
answer to 10.0.0.2

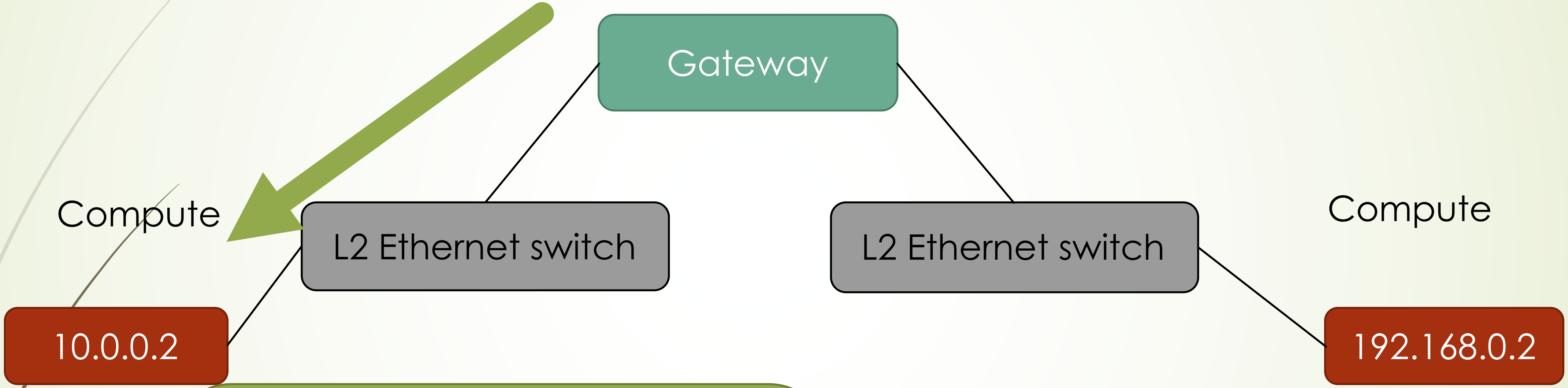
L3 – IP Routing(standard)



L2Src: MAC (192.168.0.2)
L2Dest: MAC (192.168.0.1)

ICMP reply to 10.0.0.2
from 192.168.0.2

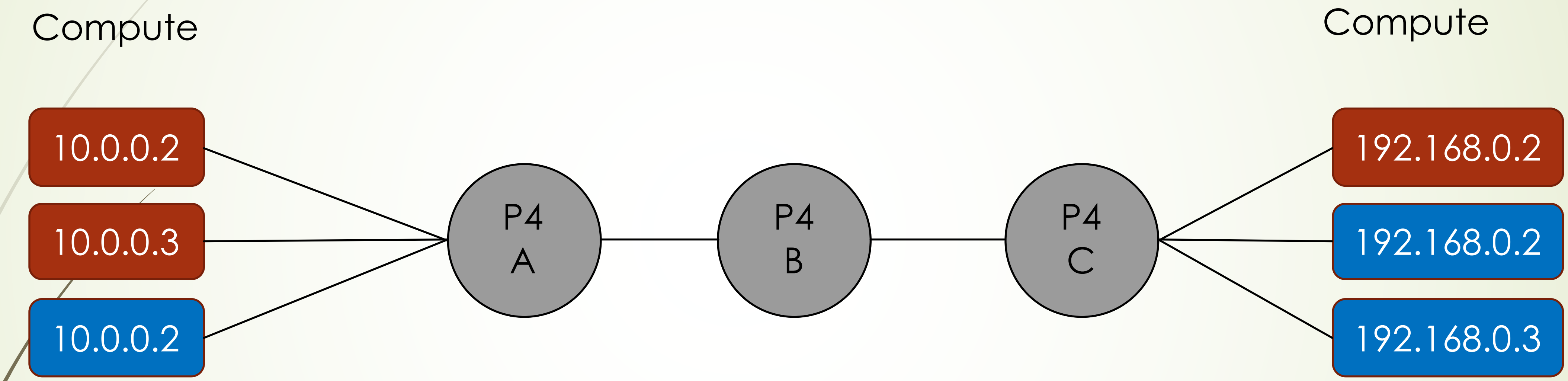
L3 – IP Routing(standard)



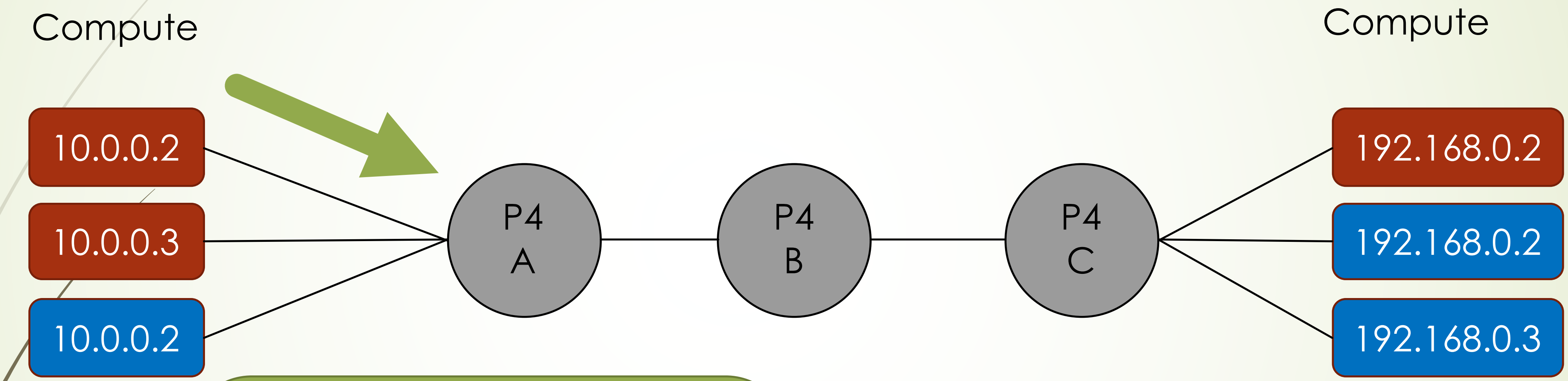
L2Src: MAC (10.0.0.1)
L2Dest: MAC (10.0.0.2)

ICMP reply to 10.0.0.2
from 192.168.0.2

L3 – IP Routing (using P4 VPC)



L3 – IP Routing (using P4 VPC)



L2Src: MAC(10.0.0.2)
L2Dest: FF:FF:FF:FF:FF:FF

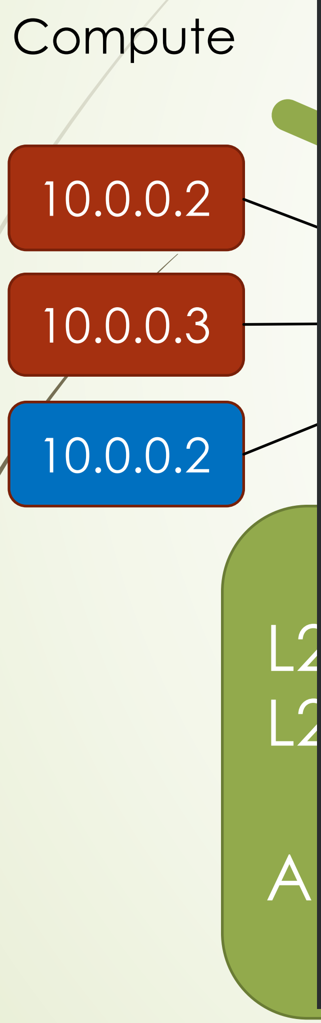
ARP who has 10.0.0.1

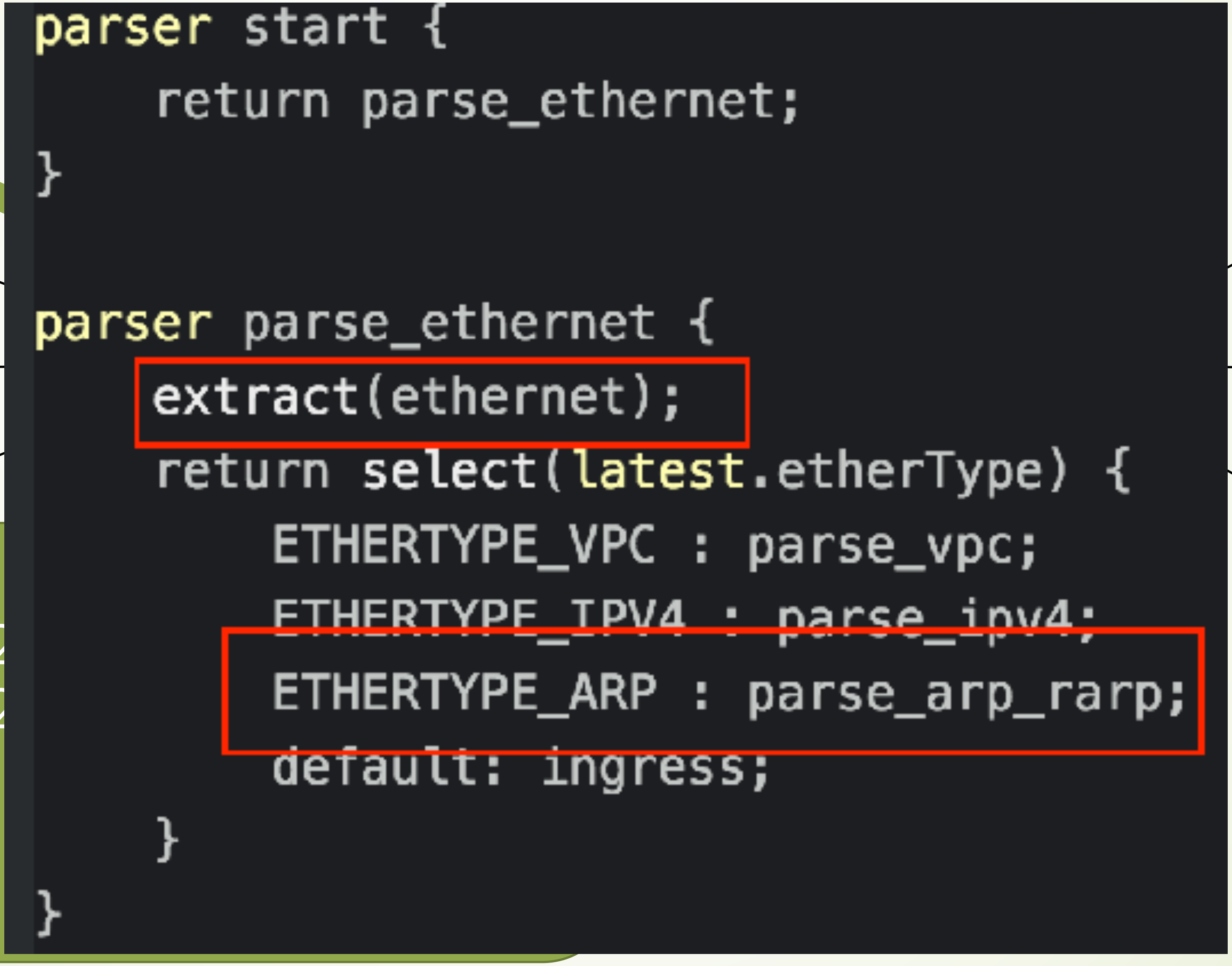
L3 – IP Routing (using P4 VPC)

Compute
Compute
10.0.0.2
192.168.0.2
ARP who has 10.0.0.1

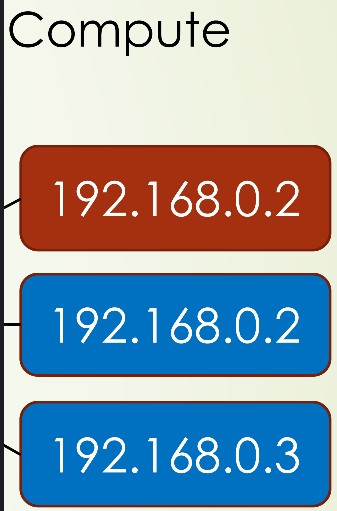
We do not have a GW in our network but we will reply a ARP reply with a fake GW MAC

L3 – IP Routing (using P4 VPC)





```
parser start {
    return parse_ethernet;
}

parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        ETHERTYPE_VPC : parse_vpc;
        ETHERTYPE_IPV4 : parse_ipv4;
        ETHERTYPE_ARP : parse_arp_rarp;
        default: ingress;
    }
}
```

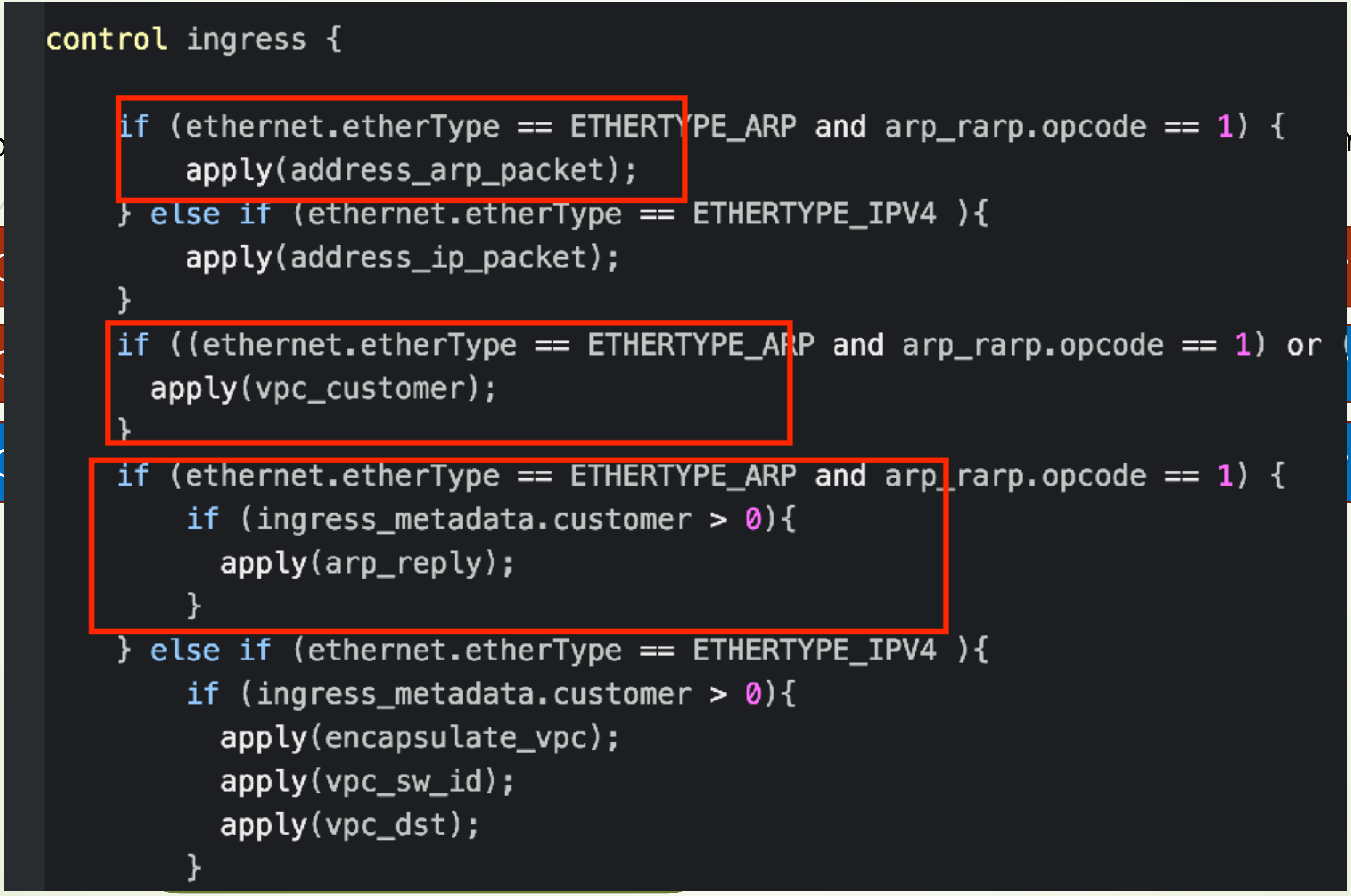


L3 – IP Routing (using P4 VPC)

Co

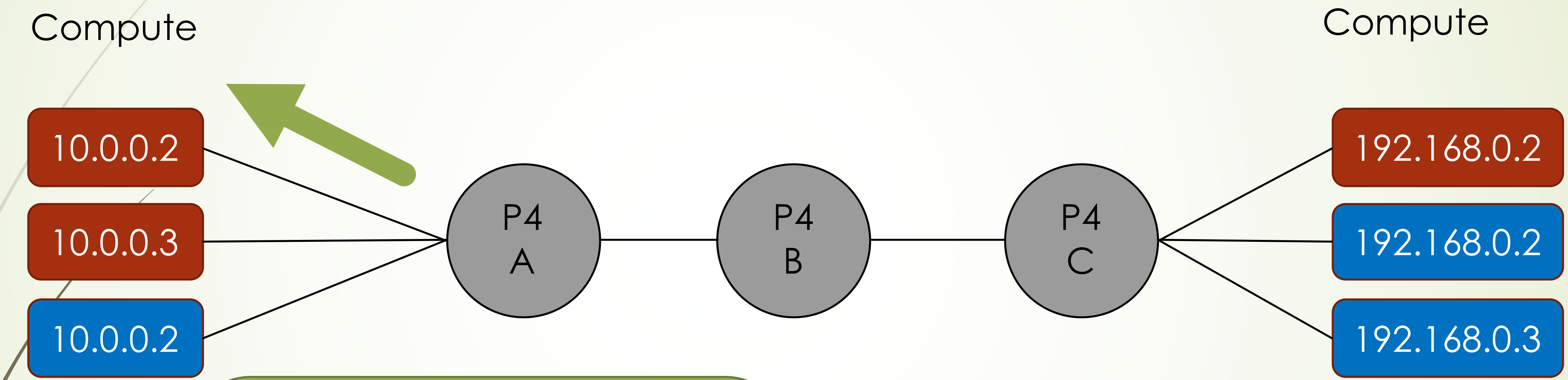
mpute

10	2.168.0.2
10	2.168.0.2
10	2.168.0.3



```
control ingress {

    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        apply(address_arp_packet);
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        apply(address_ip_packet);
    }
    if ((ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) or (
      apply(vpc_customer);
    }
    if (ethernet.etherType == ETHERTYPE_ARP and arp_rarp.opcode == 1) {
        if (ingress_metadata.customer > 0){
          apply(arp_reply);
        }
    } else if (ethernet.etherType == ETHERTYPE_IPV4 ){
        if (ingress_metadata.customer > 0){
          apply(encapsulate_vpc);
          apply(vpc_sw_id);
          apply(vpc_dst);
        }
```

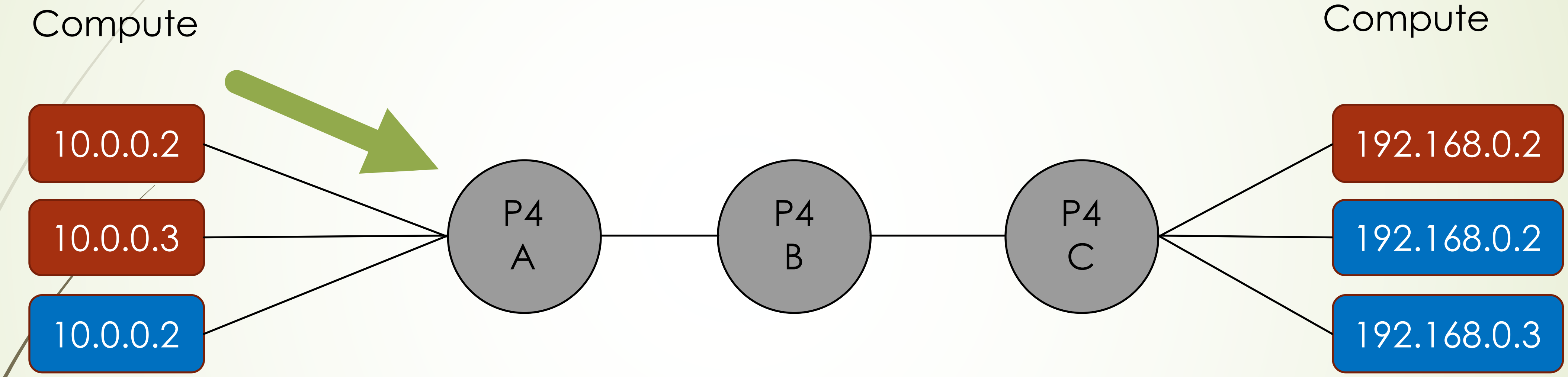
L3 – IP Routing (using P4 VPC)



L2Src: MAC(10.0.0.1)
L2Dest: MAC(10.0.0.2)

ARP I have 10.0.0.1

L3 – IP Routing (using P4 VPC)



L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.1)

ICMP to 192.168.0.2 answer to 10.0.2

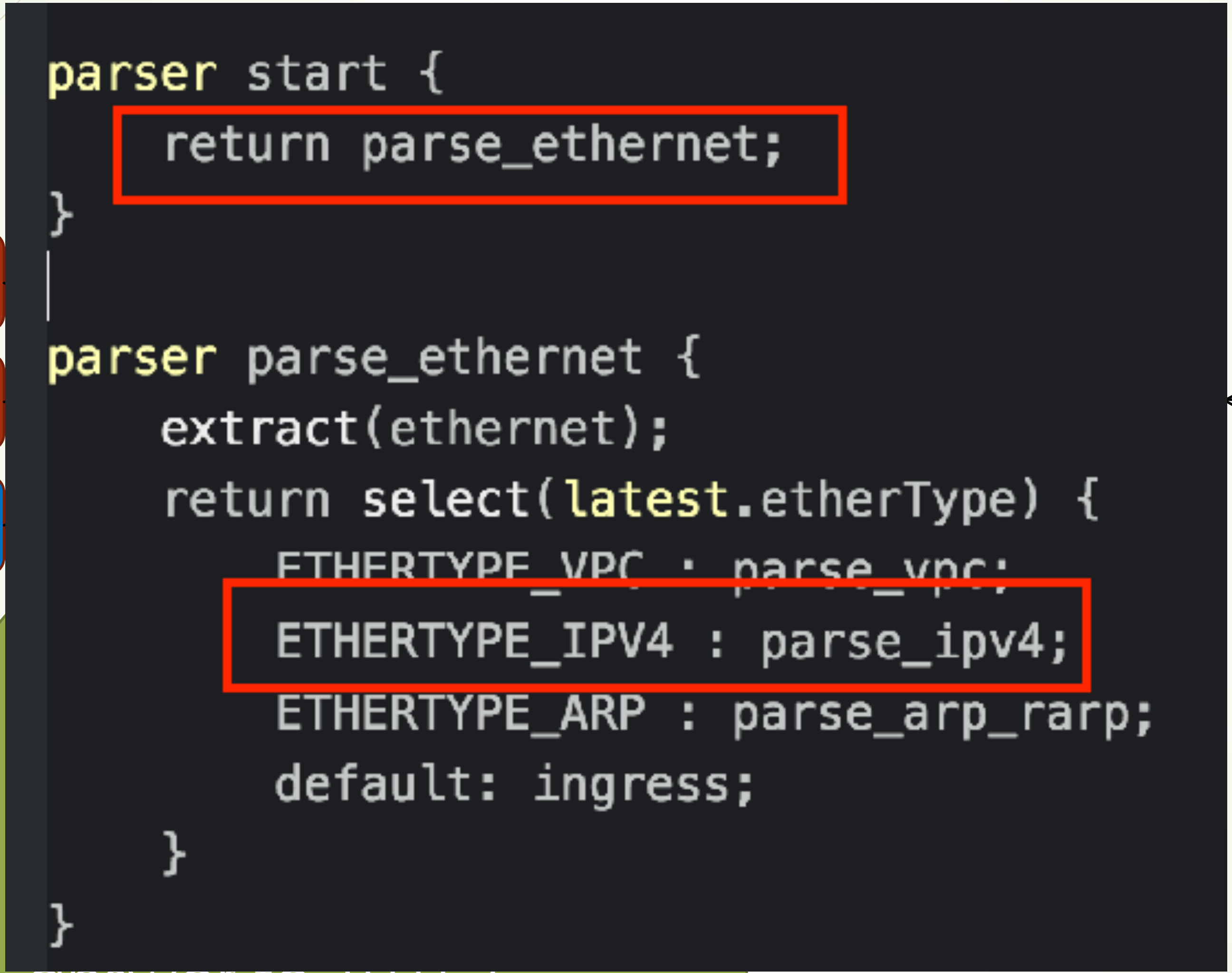
L3 – IP Routing (using P4 VPC)

Compute

10.0.0.2

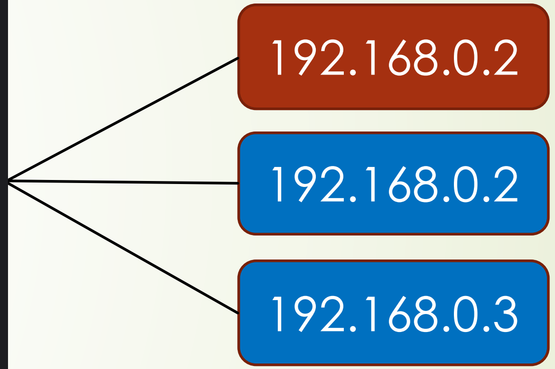
10.0.0.3

10.0.0.2



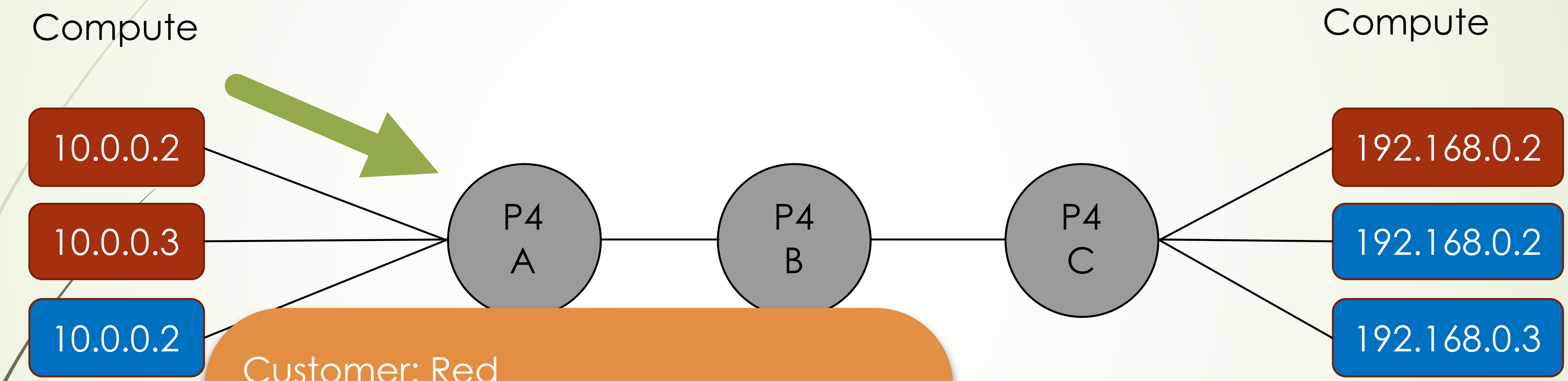
```
parser start {
    return parse_ethernet;
}

parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        ETHERTYPE_VPC : parse_vpc;
        ETHERTYPE_IPV4 : parse_ipv4;
        ETHERTYPE_ARP : parse_arp_rarp;
        default: ingress;
    }
}
```

Compute



answer to 10.0.2

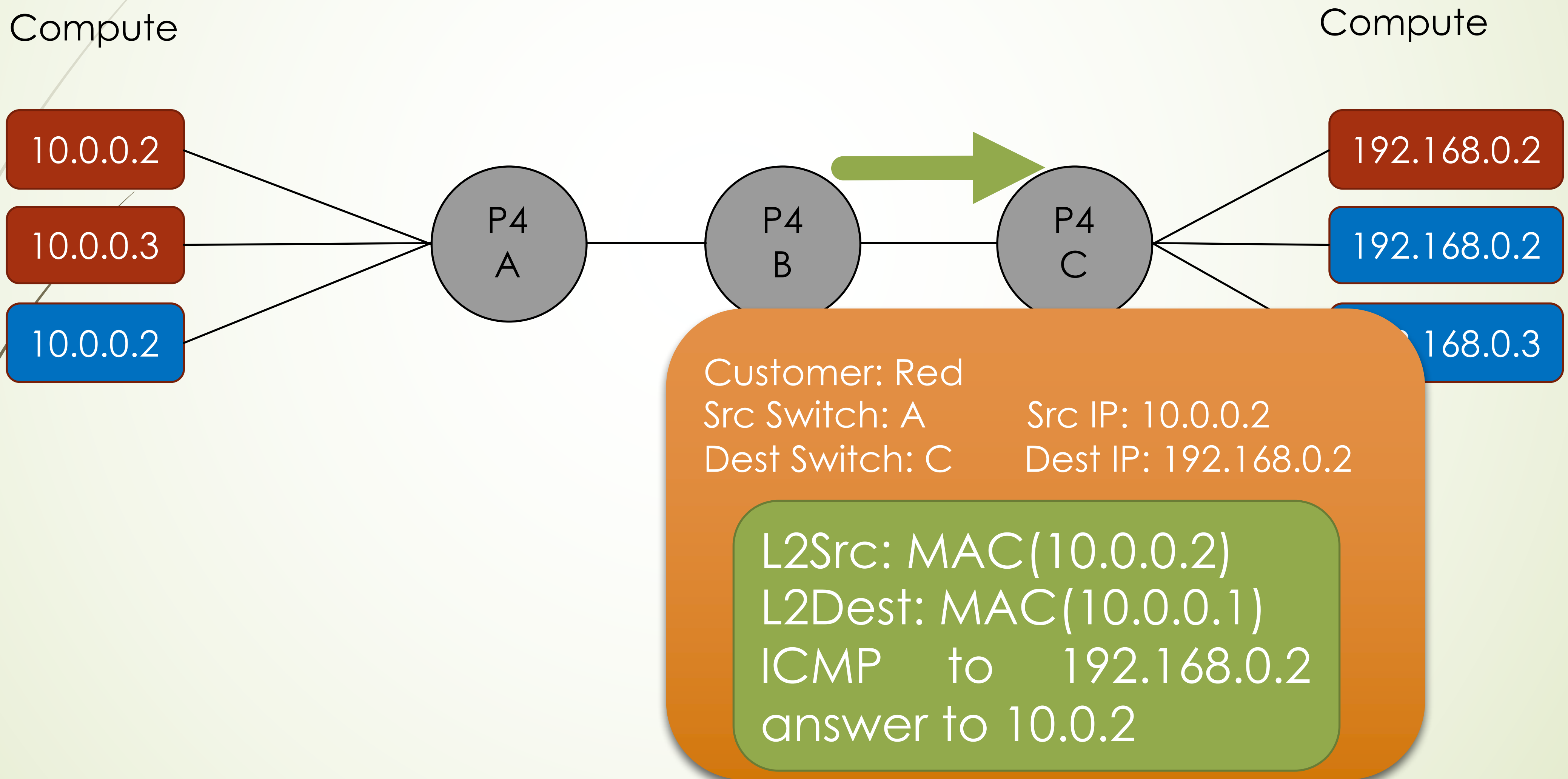
L3 – IP Routing (using P4 VPC)



Customer: Red
Src Switch: A   Src IP: 10.0.0.2
Dest Switch: C   Dest IP: 192.168.0.2

L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.1)
ICMP to 192.168.0.2
answer to 10.0.2

L3 – IP Routing (using P4 VPC)



Customer: Red
Src Switch: A
Dest Switch: C
Src IP: 10.0.0.2
Dest IP: 192.168.0.2

L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.1)
ICMP to 192.168.0.2
answer to 10.0.2

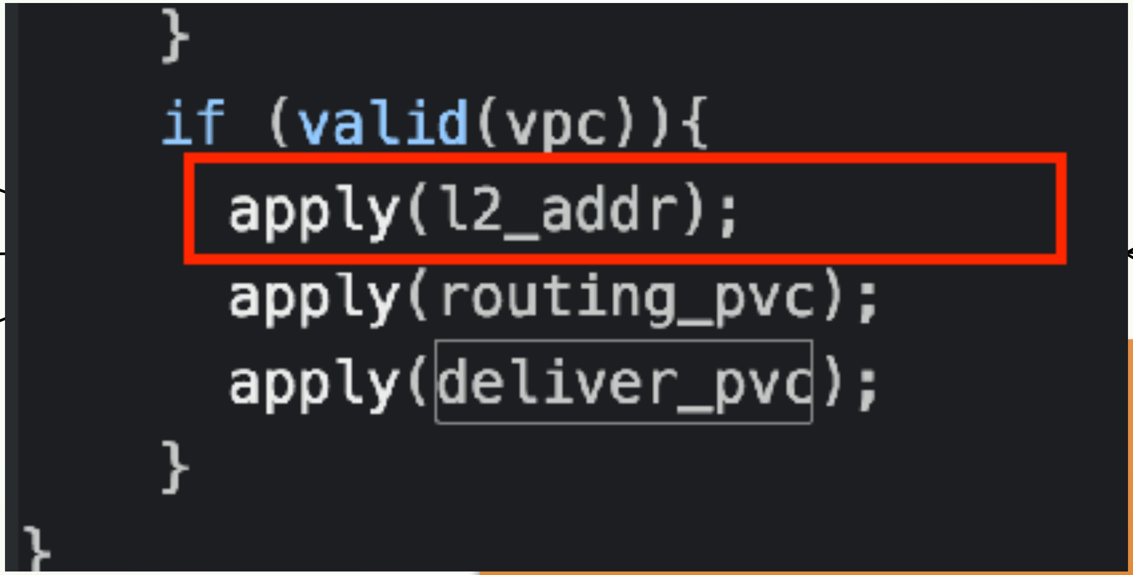
L3 – IP Routing (using P4 VPC)

Compute

10.0.0.2

10.0.0.3

10.0.0.2



```
    }
    if (valid(vpc)){
      apply(l2_addr);
      apply(routing_pvc);
      apply(deliver_pvc);
    }
}
```

Compute

192.168.0.2

192.168.0.2

168.0.3

0.0.0.2
192.168.0.2

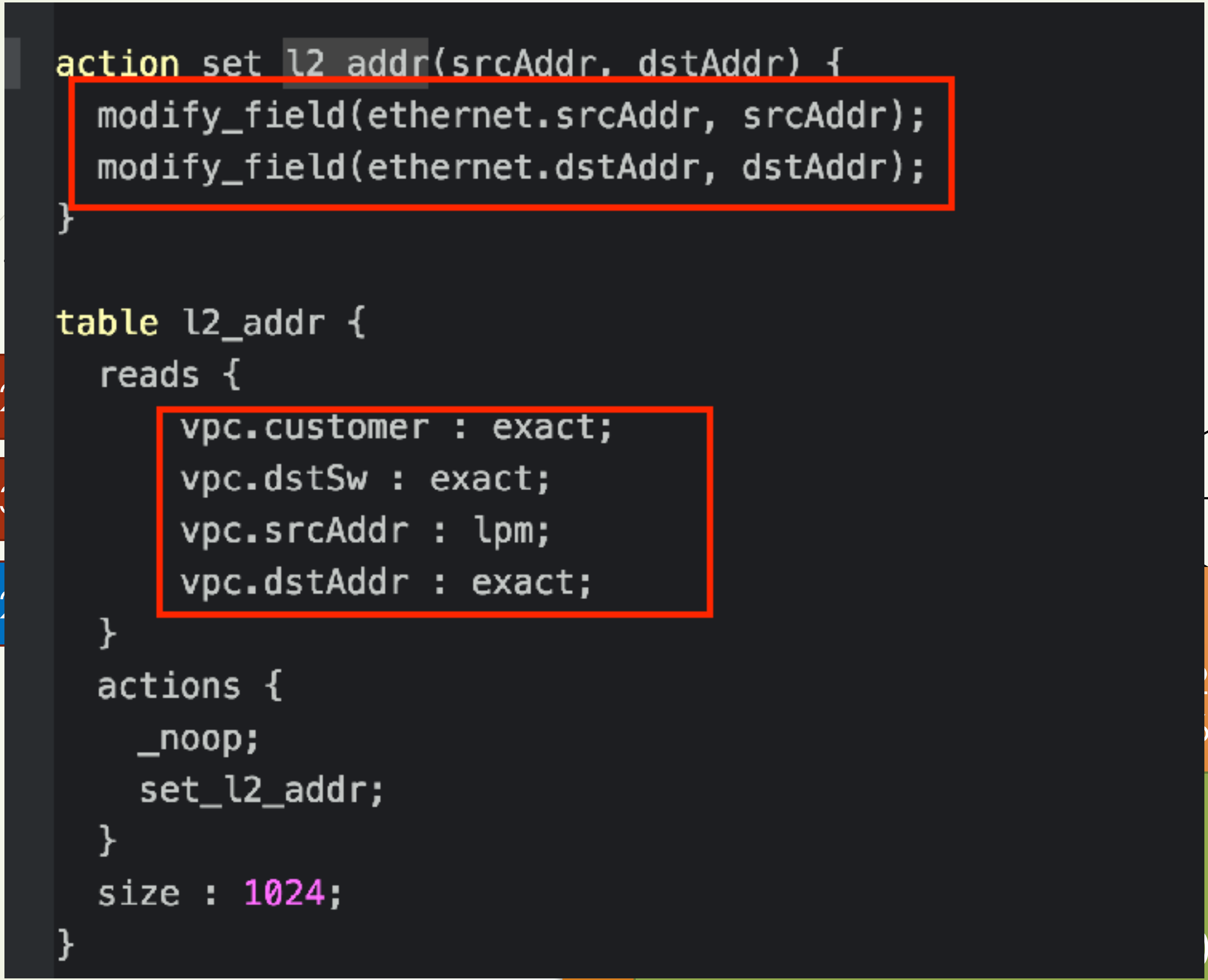
L2Src: MAC(10.0.0.2)
L2Dest: MAC(10.0.0.1)
ICMP to 192.168.0.2
answer to 10.0.2

Compu

10.0.0.2

10.0.0.3

10.0.0.2



```
action set_l2_addr(srcAddr, dstAddr) {
  modify_field(ethernet.srcAddr, srcAddr);
  modify_field(ethernet.dstAddr, dstAddr);
}

table l2_addr {
  reads {
      vpc.customer : exact;
      vpc.dstSw : exact;
      vpc.srcAddr : lpm;
      vpc.dstAddr : exact;
  }
  actions {
    _noop;
    set_l2_addr;
  }
  size : 1024;
}
```

Compute

192.168.0.2

192.168.0.2

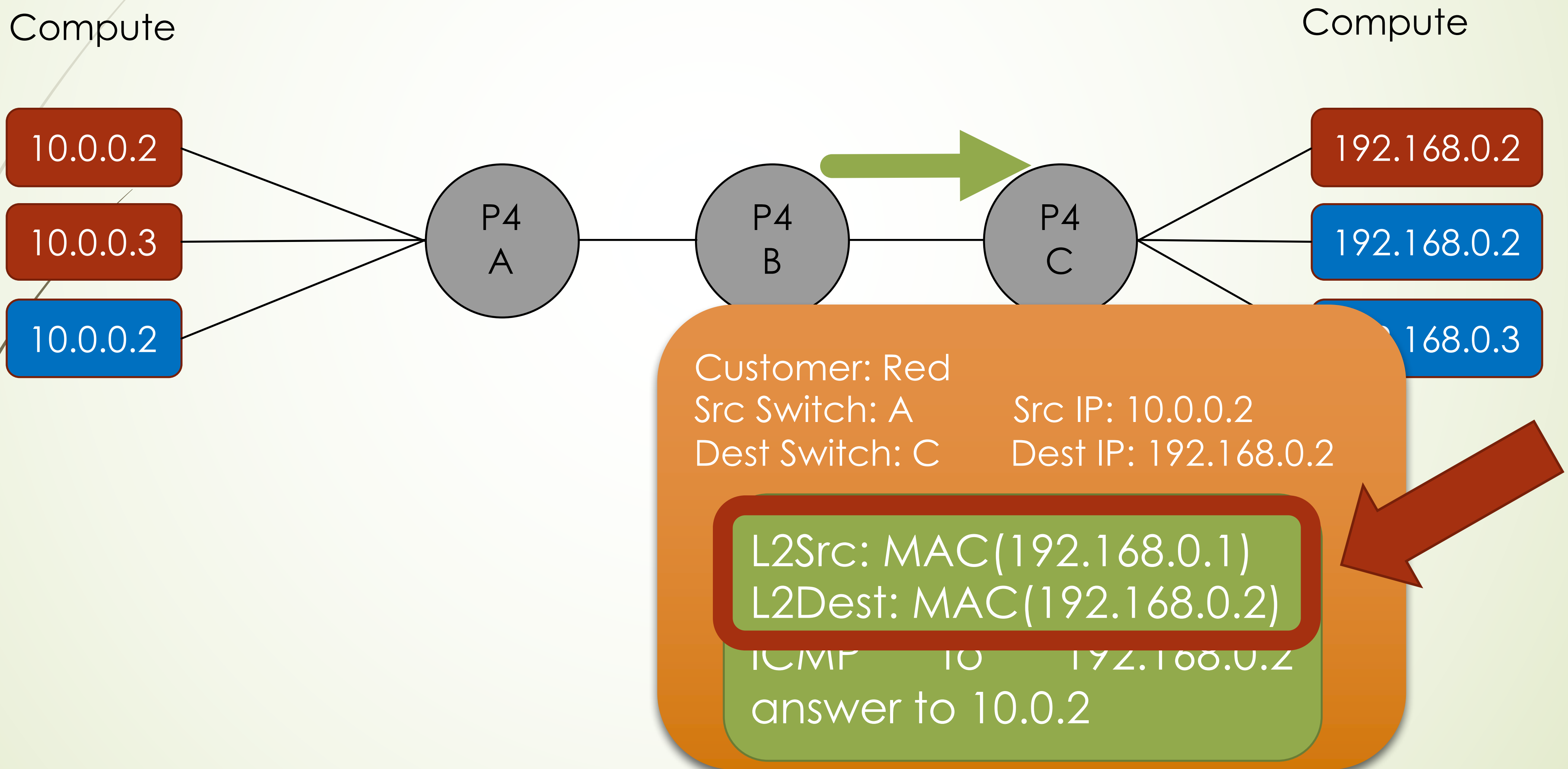
168.0.3

8.0.2

.2

answer to 10.0.2

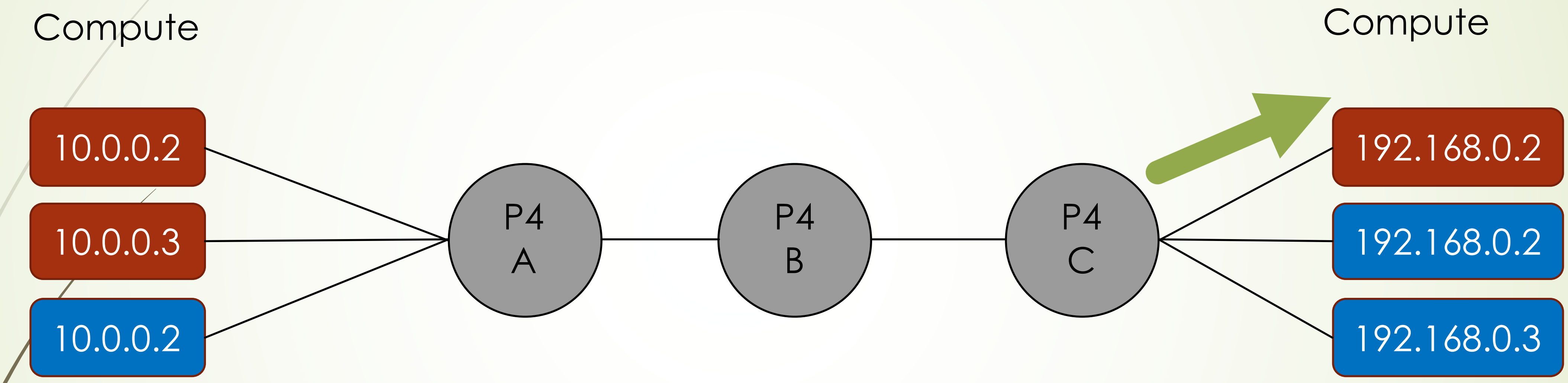
L3 – IP Routing (using P4 VPC)



Customer: Red
Src Switch: A
Dest Switch: C
Src IP: 10.0.0.2
Dest IP: 192.168.0.2

L2Src: MAC(192.168.0.1)
L2Dest: MAC(192.168.0.2)
ICMP To 192.168.0.2
answer to 10.0.2

L3 – IP Routing (using P4 VPC)



L2Src: MAC(192.168.0.1)
L2Dest: MAC(192.168.0.2)
ICMP to 192.168.0.2 answer to 10.0.2

Summary

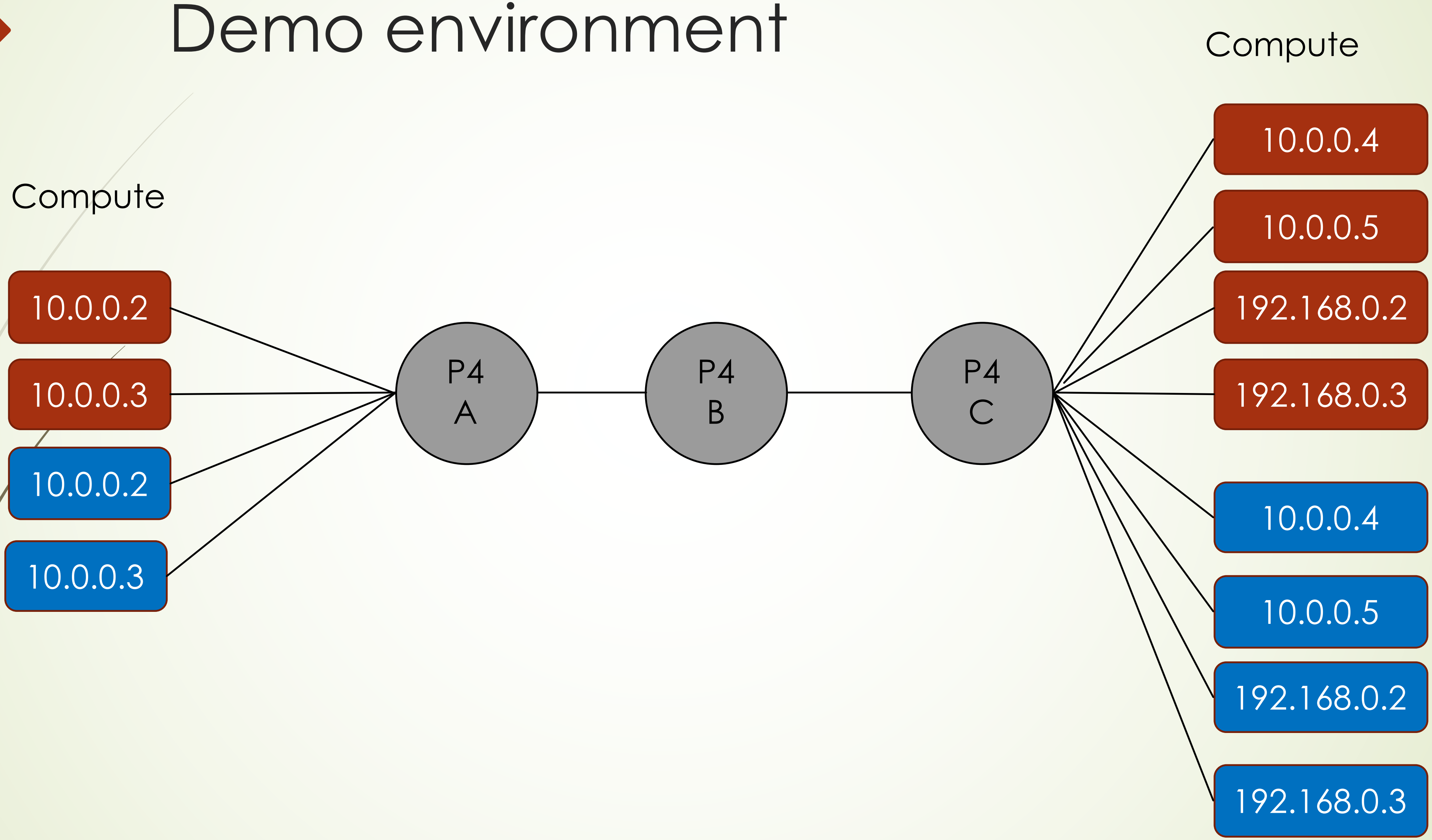
- Custom parser that just covers ARP, IP and VPC headers
- Create a new encapsulation using 0x0777 Ethernet protocol VPC (custom and non standard with fields like customer id, src/destination IP Address, …)
- Manipulation of packets:
  - Convert an ARP request into an ARP reply
  - Encapsulate IP packets with VPC (0x0777)
  - Route based on VPC header
  - Modify any field in the packet header
- Simulate a non existing GW with a custom protocol

Demo

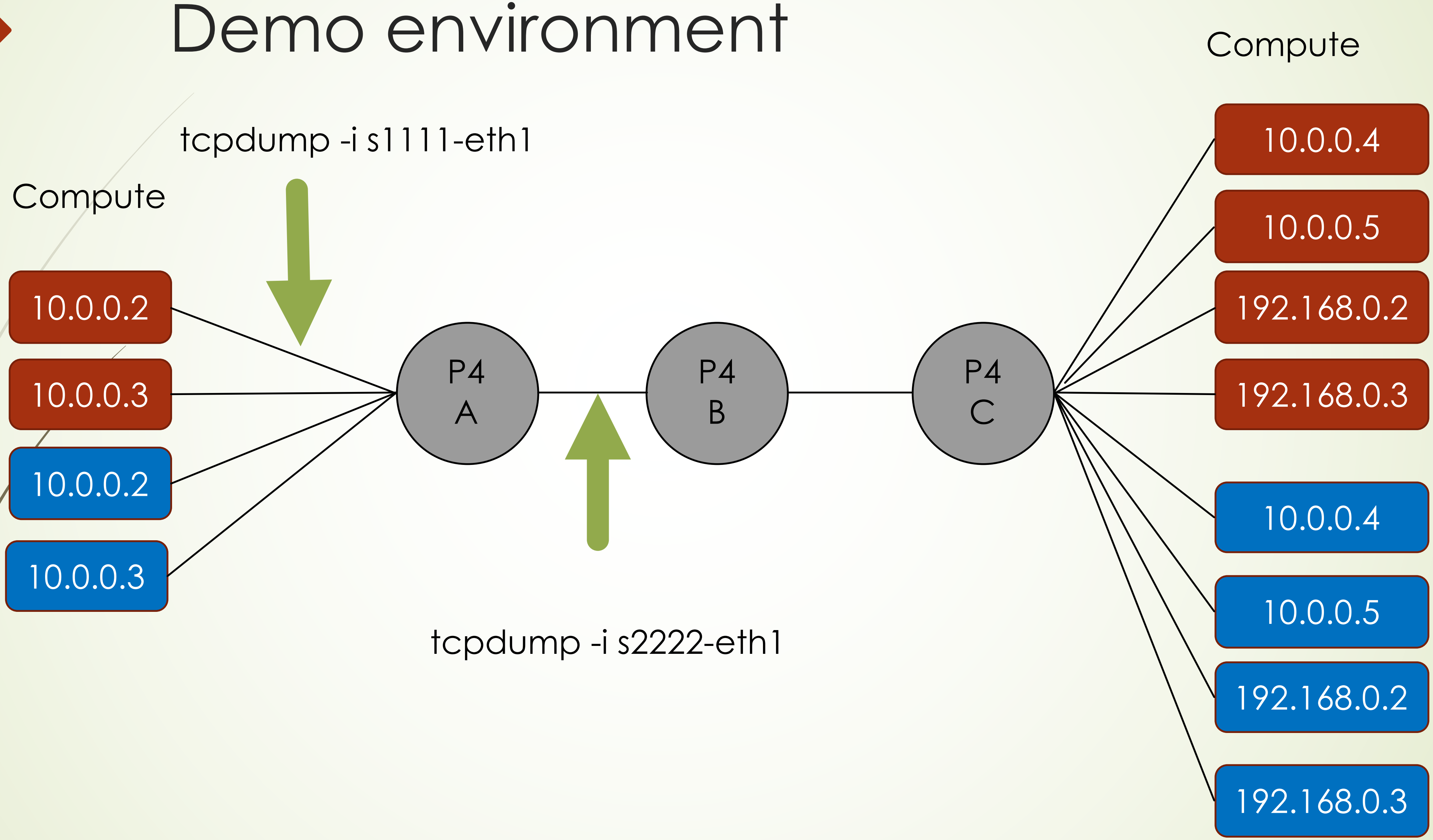
Steps

- Ping from red customer to an ip address in the same subnet.
  - h102red ping 10.0.0.4
  - h102red nc 10.0.0.4 8888
  - h104red ifconfig eth0
- Ping from blue customer to same ip address as previous test
  - h102blue ping 10.0.0.4
  - h102blue nc 10.0.0.4 8888
  - h103blue ifconfig eth0
- Check same test with different ip subnet 192.168.0.3 h203blue or h203red

Demo environment



Demo environment



Compute
tcpdump -i s1111-eth1
Compute
10.0.0.4
10.0.0.5
192.168.0.2
192.168.0.3
10.0.0.2
10.0.0.3
10.0.0.2
10.0.0.3
P4 A
P4 B
P4 C
10.0.0.4
10.0.0.5
192.168.0.2
192.168.0.3
tcpdump -i s2222-eth1