

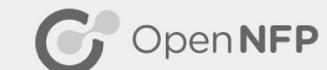
A photograph of a person walking down a modern staircase with glass railings. The person is seen from behind, wearing dark clothing. The stairs are made of dark metal and glass, and the background shows a bright, modern interior space.

NETRONOME

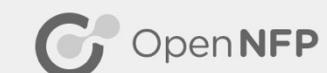
P4 Introduction

Jaco Joubert
SIGCOMM 2018

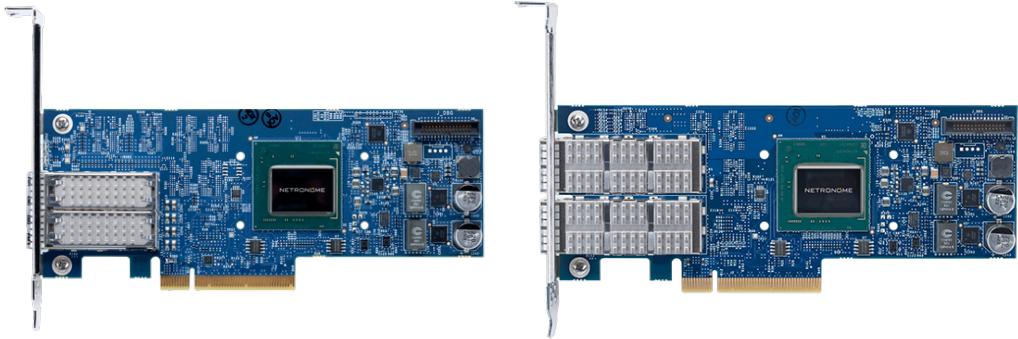
- ▶ Introduction
- ▶ Learning Goals
- ▶ Agilio SmartNIC
- ▶ Process to Develop Code
- ▶ P4/C Examples
 - P4 INT
 - P4/C Stateful Firewall
- ▶ SmartNIC P4 Performance
- ▶ Demo Setup
 - P4 INT
 - P4C -> XDP
- ▶ Introduction to P4
 - P4-16 Code
- ▶ Learn More



- ▶ Overview of the Netronome P4 SDK capabilities
- ▶ Integrated Development Environment
 - Elements - Projects, Files
 - Workflows - Editing, Debugging
- ▶ Introduction to P4-16
- ▶ P4 Constructs
 - Parsing, Matching, Actions
- ▶ Simple Switch Architecture Overview
- ▶ P4C-XDP Architecture Overview
- ▶ Live Demo: See It Run
 - Q&A



- ▶ 1x 10/40GbE
- ▶ 2x 10/40/25GbE
- ▶ Fully Programmable
 - Micro-C
 - P4
 - P4 + Sandbox C
- ▶ Software Development Kit (SDK)
 - Programmer Studio (IDE)
 - Runtime Environment (Server Side)
 - Command Line Tools



Agilio® CX SmartNICs



Agilio® Software Solutions





► The 4 Ps of P4:

- Protocol Independent Packet Processing Programming
- Programmer defines packet formats and processing algorithms

► Target Independence

- No knowledge of low-level hardware organization is required
- Compiler compiles the program for the target device
- Architecture dependent - e.g. v1model.p4, p4c-xdp.p4, psa.p4

► Consistent Control Plane Interface

- Control Plane APIs are automatically generated by the compiler

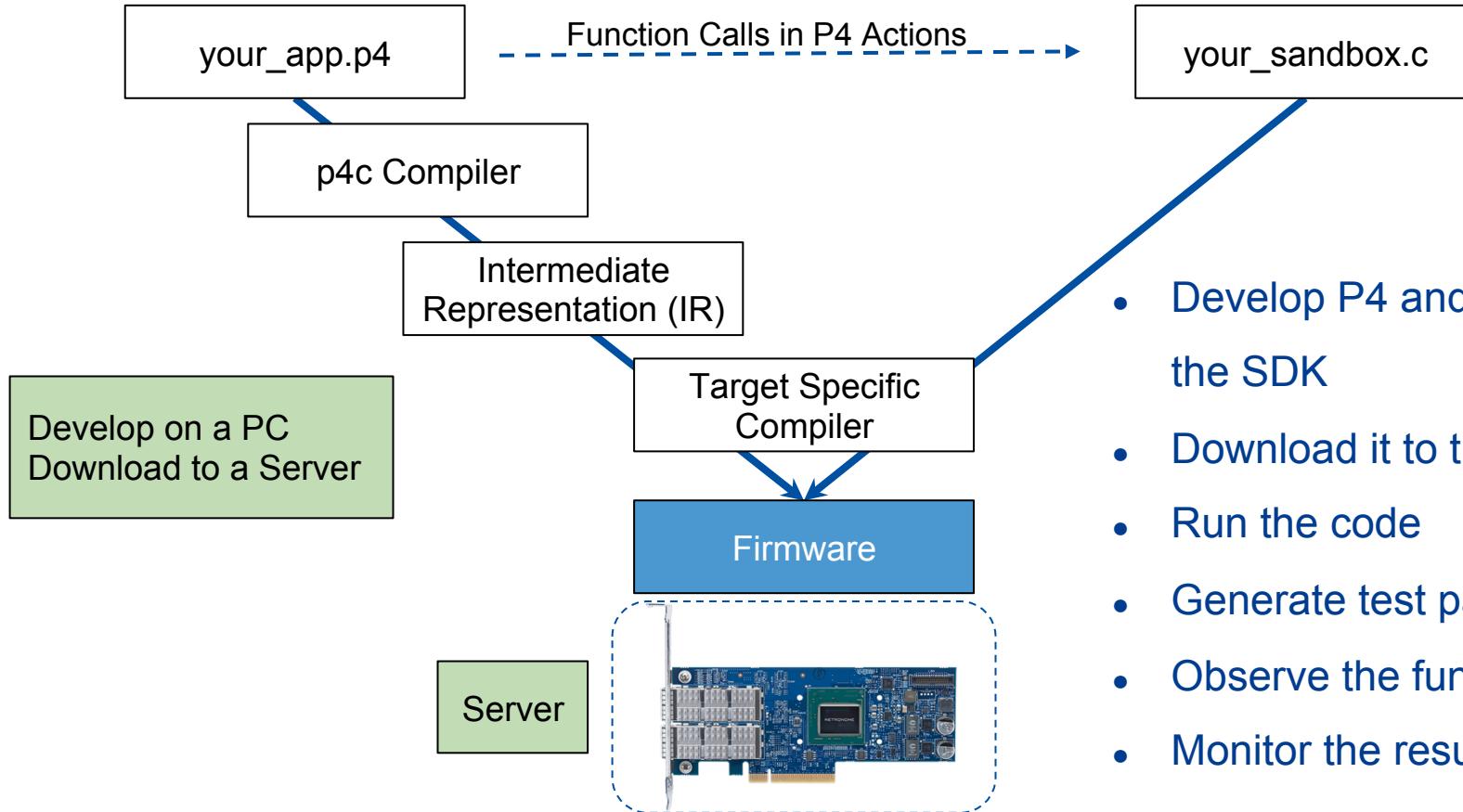
► Reconfigurability

- Data Plane program can be changed in the field

► Community-driven Design

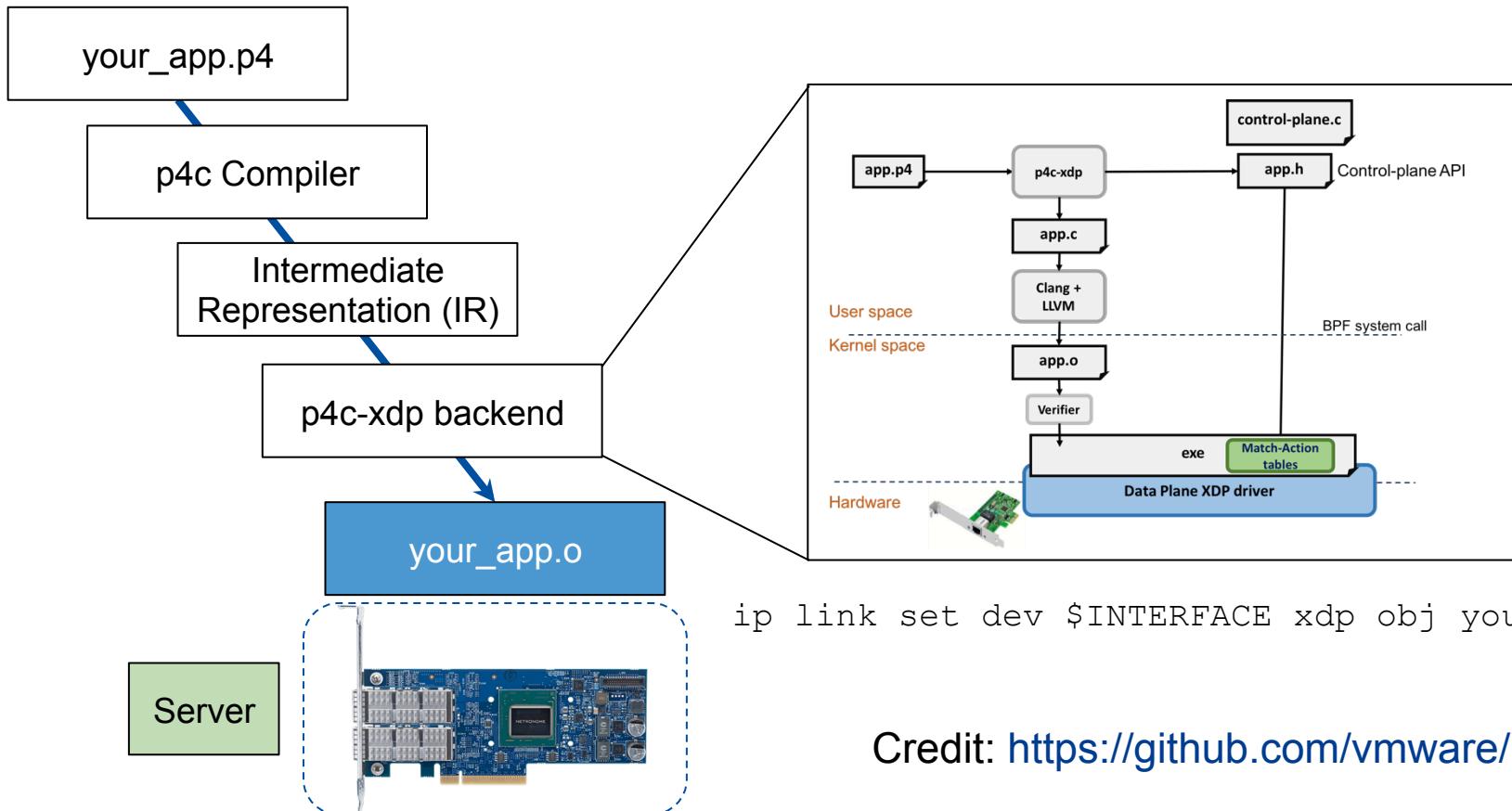
- <http://p4.org>



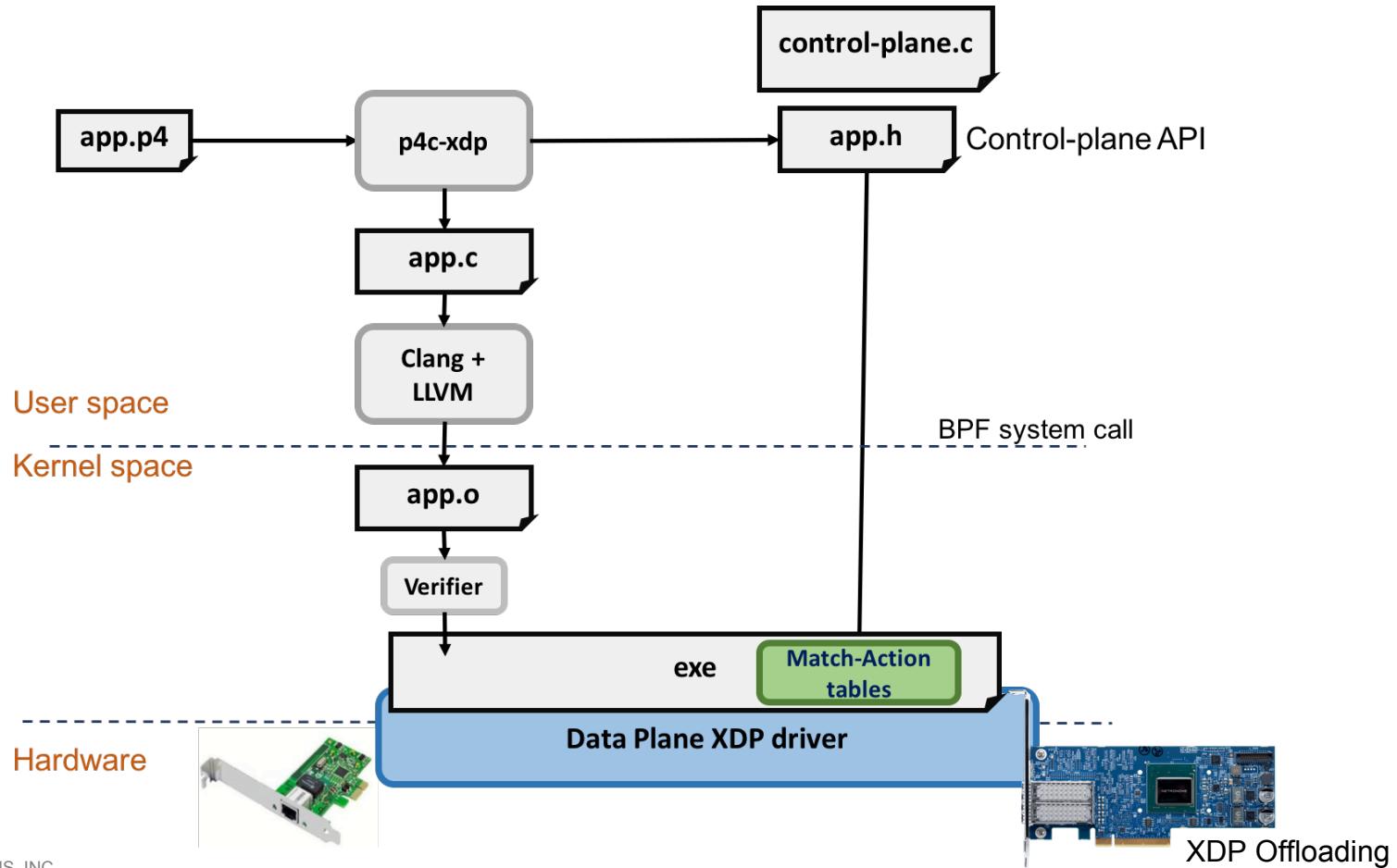


- Develop P4 and C code in the SDK
- Download it to the card
- Run the code
- Generate test packets
- Observe the functionality
- Monitor the results

P4C-XDP Development Process



Credit: <https://github.com/vmware/p4c-xdp>



► In-Band Network Telemetry

- Add headers at each hop
- Take timestamps
- Calculate latencies
- Gather stats
- Remove headers and forward unchanged packet

► Stateful Firewall

- Closer look in next slides

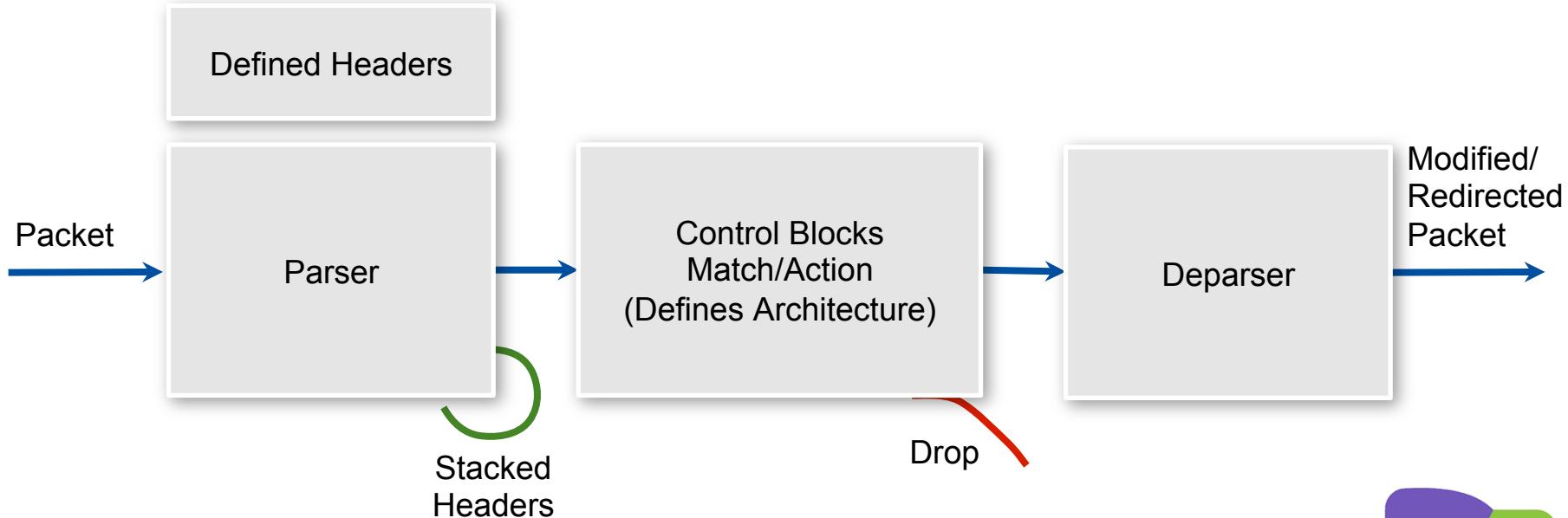
► Load Balancing

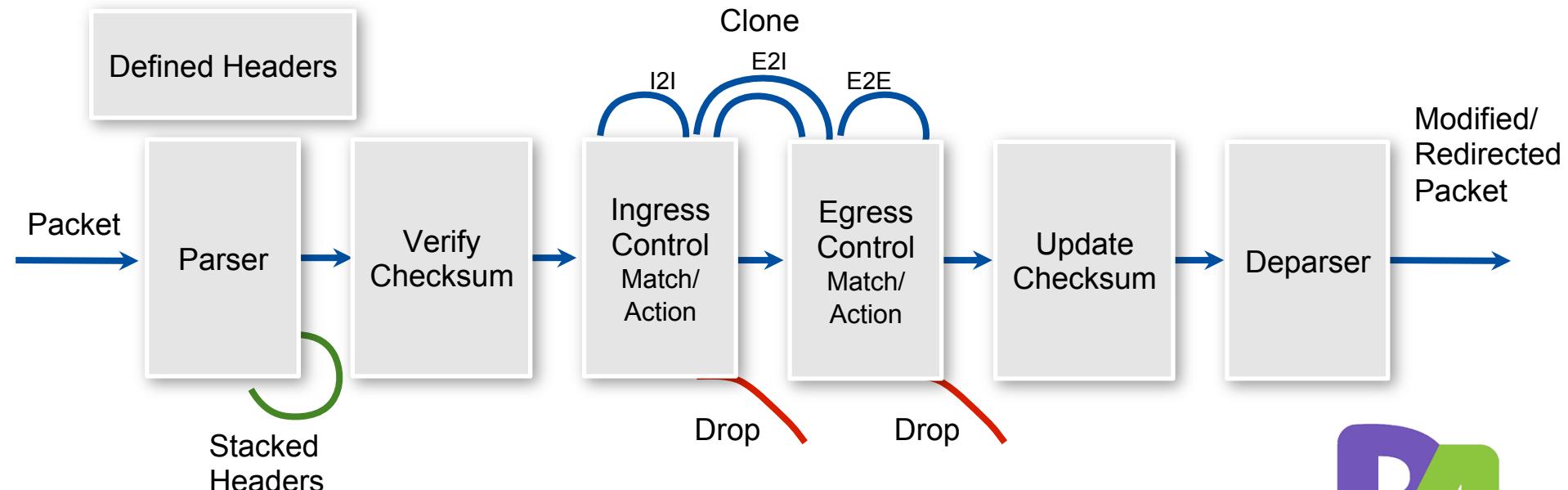
- Load balance based on hash calculated

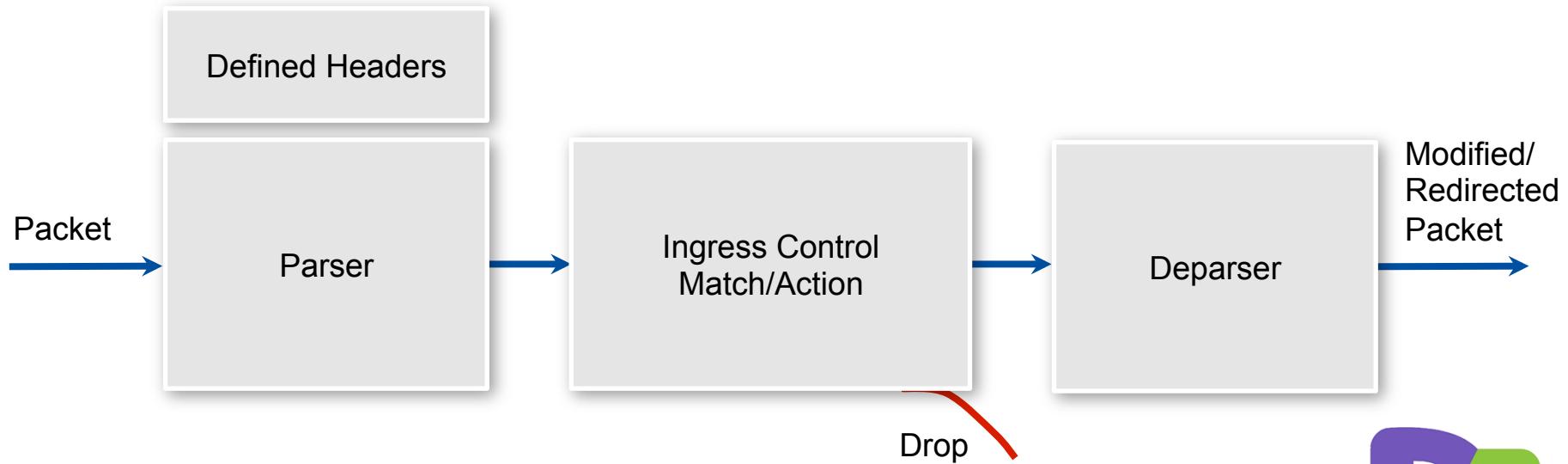
► VLAN

- Encap/Decap Headers









Model Template

```
package V1Switch<H, M>(Parser<H, M> p,  
                         VerifyChecksum<H, M> vr,  
                         Ingress<H, M> ig,  
                         Egress<H, M> eg,  
                         ComputeChecksum<H, M> ck,  
                         Deparser<H> dep  
);
```

Implementation:

```
#include "v1model.p4"  
  
package V1Switch(MyParser(),  
                 MyVerifyChecksum(),  
                 MyIngress(),  
                 MyEgress(),  
                 MyComputeChecksum(),  
                 MyDeparser())  
main;
```



```
header ethernet_t {  
    bit<48> dstAddr;  
    bit<48> srcAddr;  
    bit<16> etherType;  
}  
  
header meta_t {  
    bit<8> ioam_cnt;  
    bit<1> decapped; /* set when decapsulation is done */  
    bit<1> is_transit; /* is the packet treated as just transit */  
    bit<32> device_id; /* retrieved from a p4 register */  
}  
  
struct headers_t {  
    ethernet_t ethernet;  
    ipv4_t ipv4;  
    udp_t udp;  
}  
  
struct metadata_t {  
    meta_t meta;  
    intrinsic_metadata_t intrinsic_metadata;  
}
```



```
parser MyParser(packet_in packet,
                out    headers_t hd,
                inout metadata_t meta,
                inout standard_metadata_t standard_metadata) {

state start {
    packet.extract(hd.ethernet);
    transition select(hd.ethernet.etherType) {
        ETHERTYPE_IPV4: parse_ipv4;
        default: accept;
    }
}
```



```
control MyVerifyChecksum(  
    inout headers_t    hdr,  
    inout metadata_t   meta)  
{  
    apply {      }  
}
```

```
control MyComputeChecksum(  
    inout headers_t    hdr,  
    inout metadata_t   meta)  
{  
    apply {      }  
}
```



```
control MyIngress(inout headers_t hdr,  
                  inout metadata_t meta,  
                  inout standard_metadata_t standard_metadata)  
{  
    // Tables  
    // Actions  
    apply {  
        // 'Fixed' logic and apply tables  
        if(hdr.ipv4.isValid()) {  
            table_name.apply()  
        }  
    }  
}
```

- ▶ **Tables - Match/Actions**
 - Define behavior framework
 - Configurable behavior during runtime
 - Add Rules
- ▶ **if, else, else if**
 - 'Fixed' logic

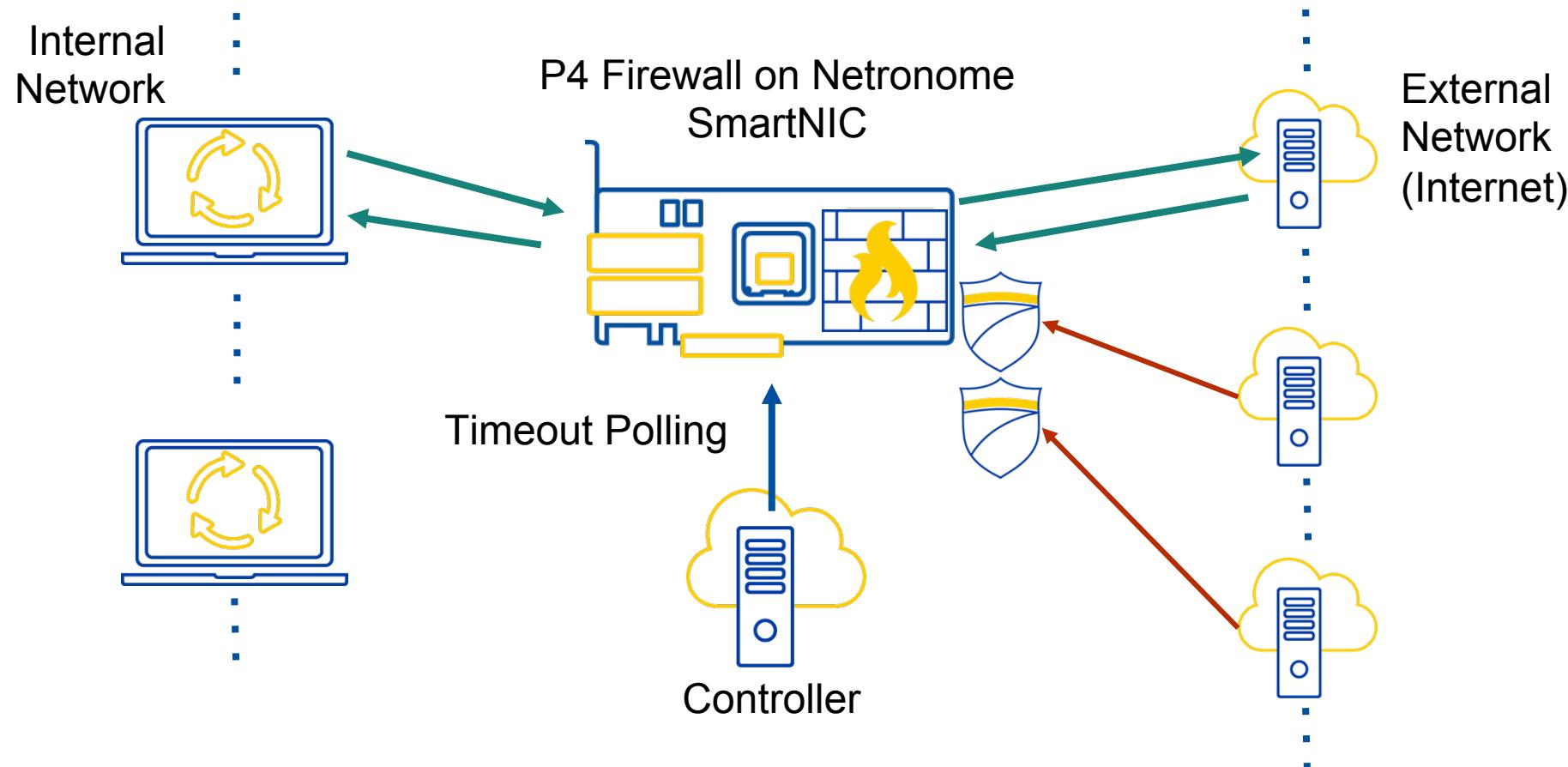


```
action do_forward(bit<16> espec, bit<1> is_transit, bit<32> device_id) {  
    meta.meta.device_id = device_id;  
    standard_metadata.egress_spec = espec;  
    meta.meta.is_transit = is_transit;  
}  
  
table tbl_forward {  
    key = {  
        standard_metadata.ingress_port : exact;  
    }  
    actions = {  
        do_forward;  
    }  
}
```

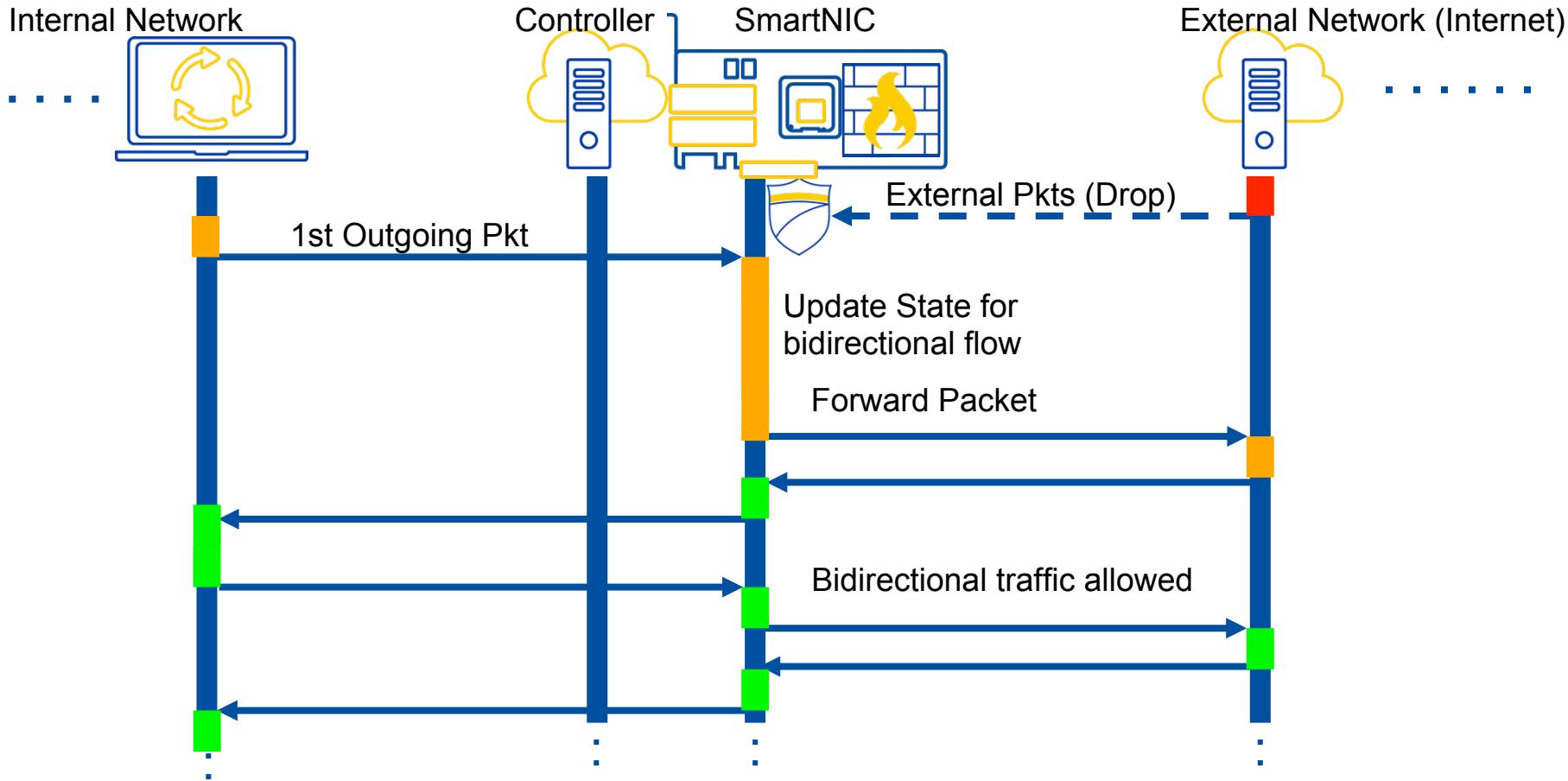


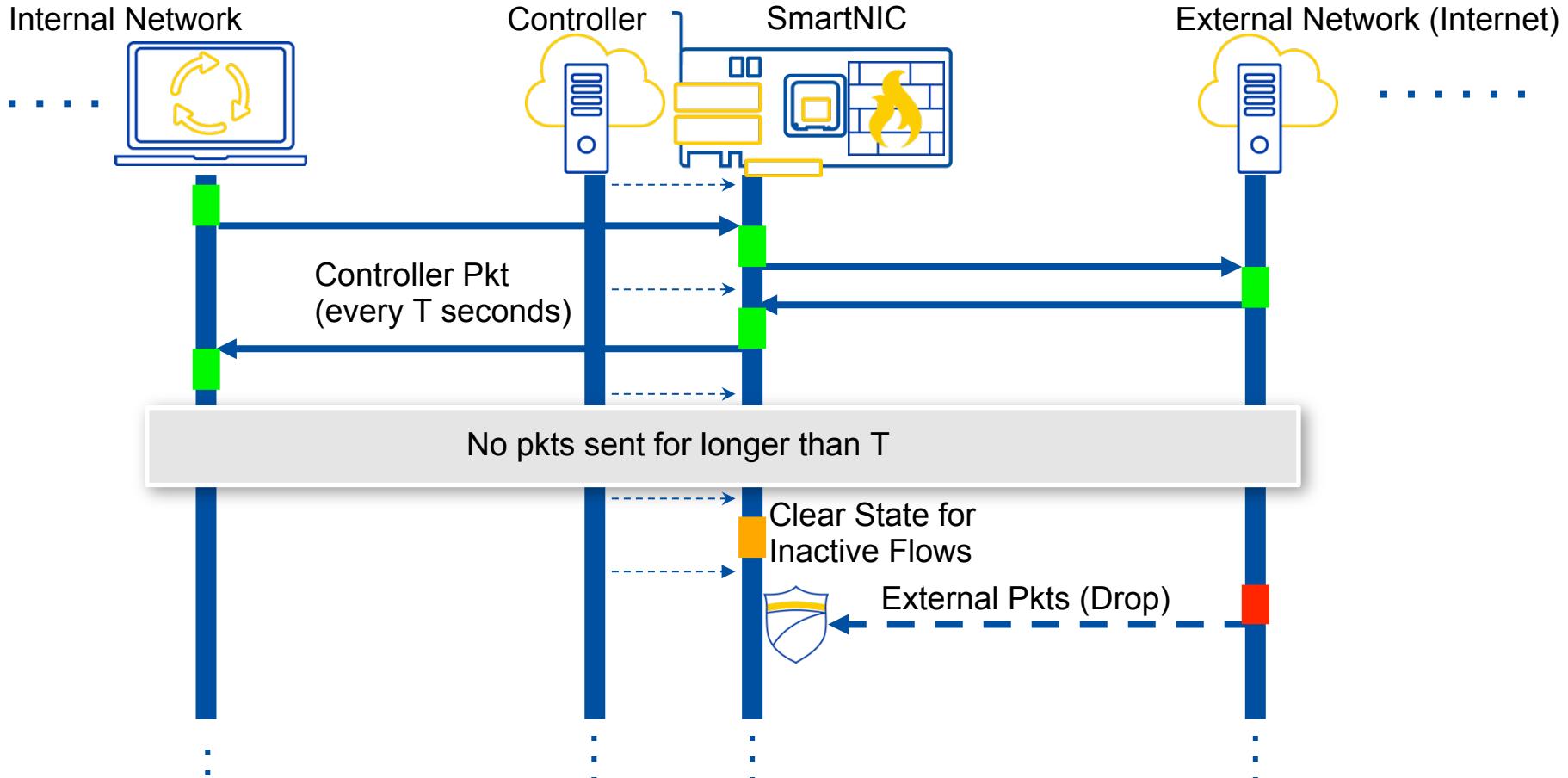
```
control MyDeparser(packet_out packet,
                     in headers_t hdr)
{
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.ipv4);
        packet.emit(hdr.udp);
        packet.emit(hdr.tcp);
    }
}
```





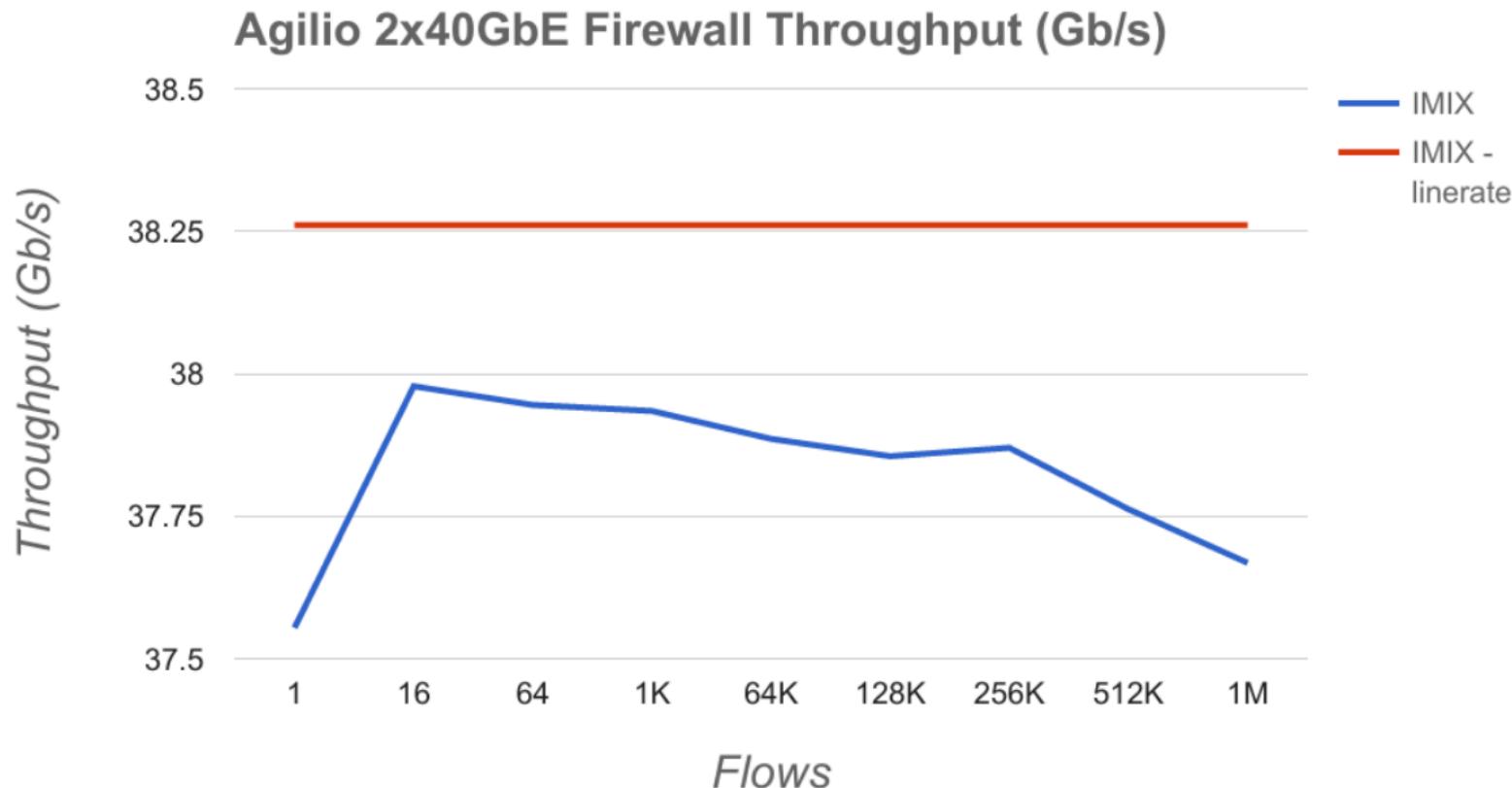
Firewall Sequence Diagram

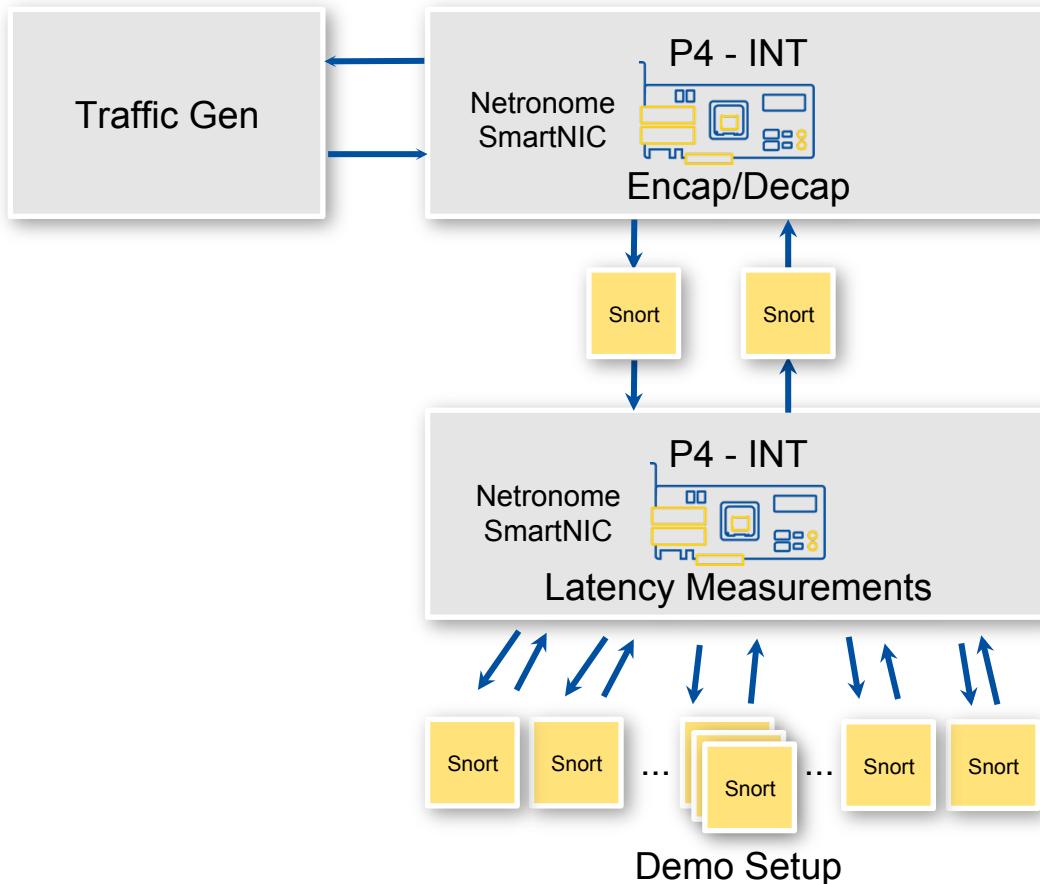




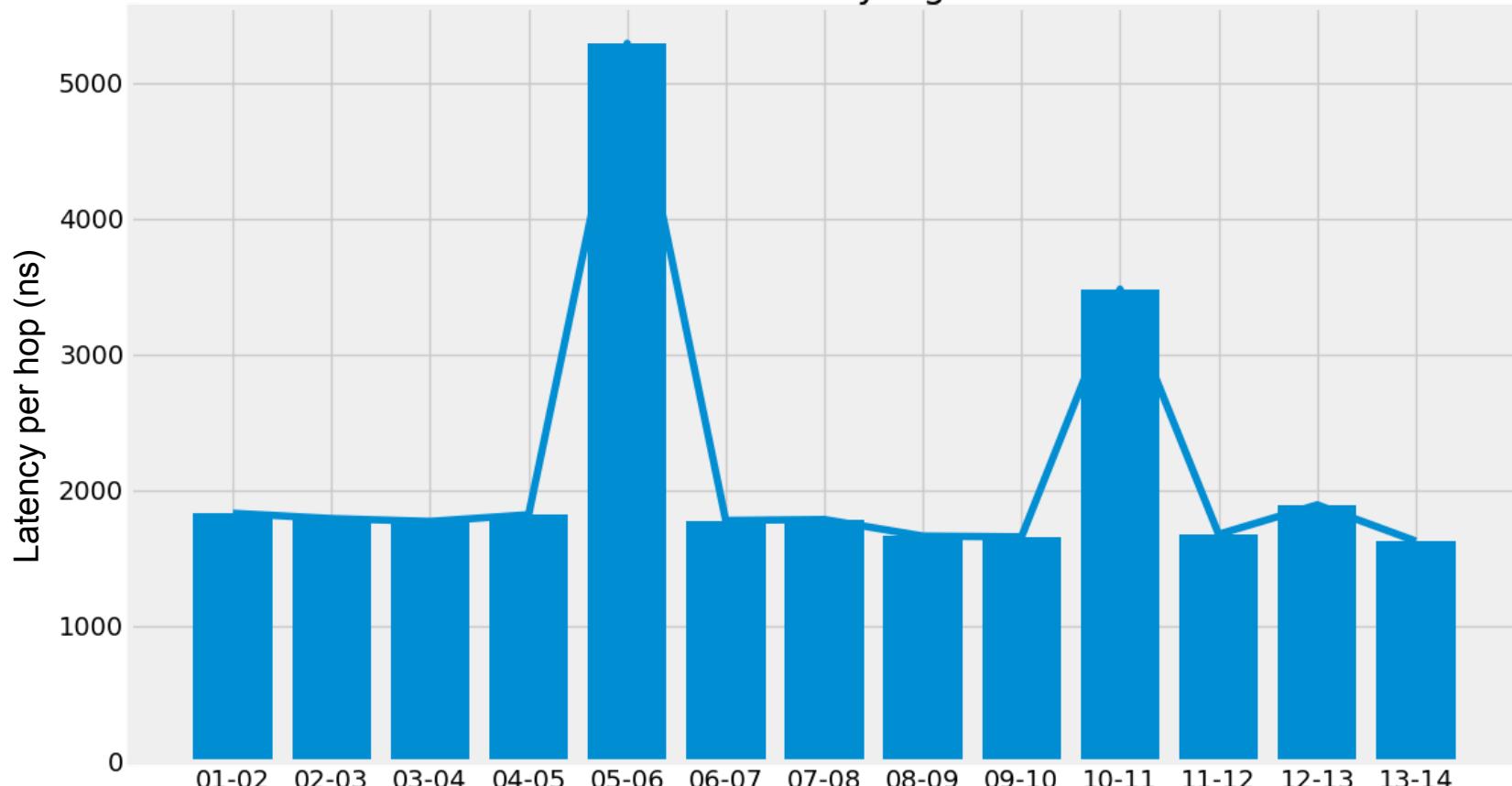
- ▶ **Stateful Forwarding on the Data Plane**
- ▶ **Sandbox C Function Calls in P4**
 - Primitive Actions
 - Hashtable implementation in Micro-C
 - Assigning and Managing Public Ports
 - Updating state
 - Dynamically clearing state for inactive flows on timeouts
- ▶ **Network Address Translation in P4**
- ▶ **Python Controller**
 - Assign available public IPs used for NAT
 - Set timeout interval
 - Send timeout packets to trigger Micro-C timeout function on the SmartNIC
- ▶ **Re-Calculate Checksums in P4**

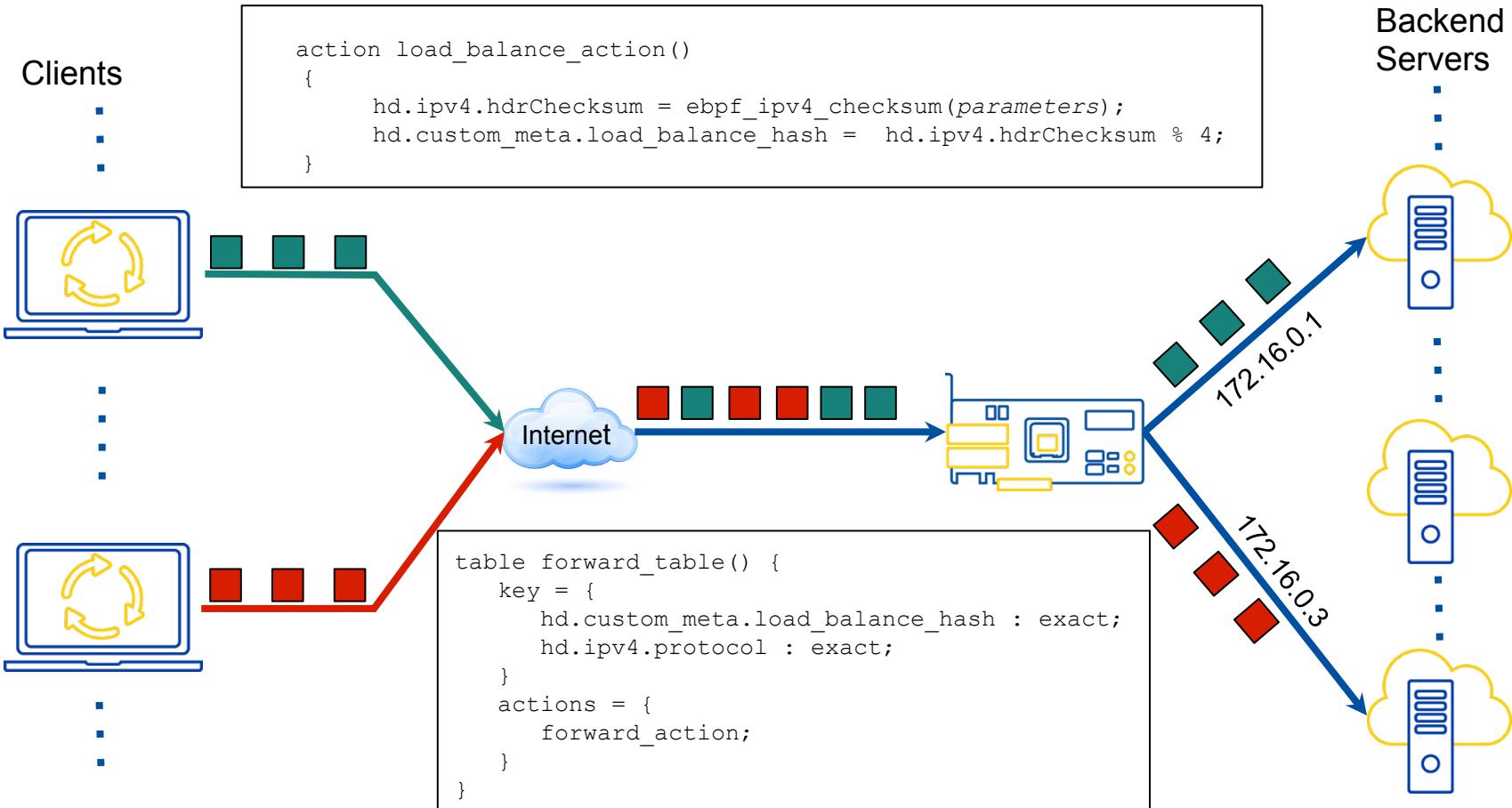


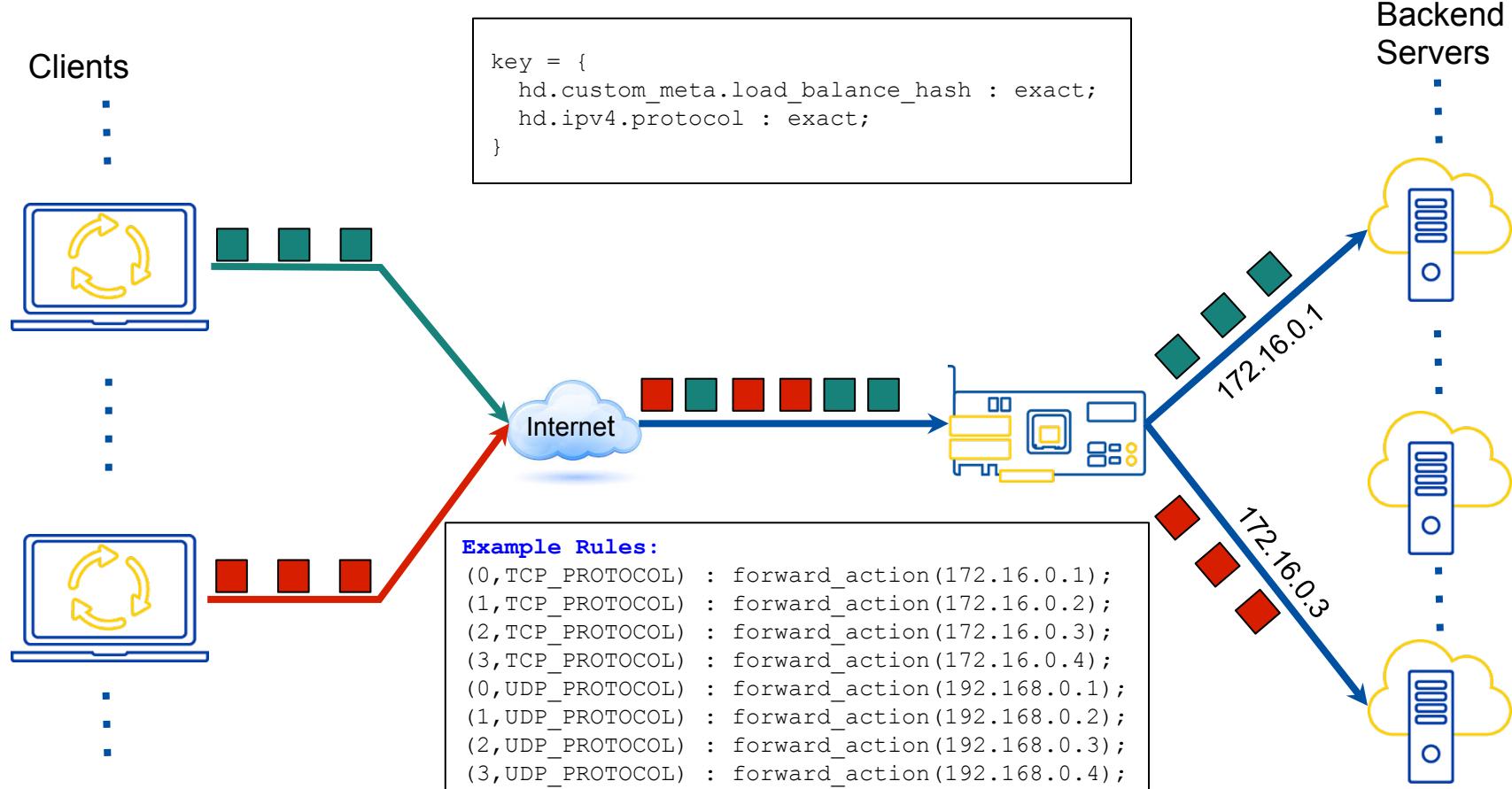




Snort Latency Figures







Original Packet:

```
15:00:43.044709 IP 10.0.0.1.3000 > 10.0.0.100.4000:  
 0x0000: ffff ffff ffff ffff ffff 0800 4500  
 0x0010: 002c 0001 0000 4011 665c 0a00 0001 0a00  
 0x0020: 0064 0bb8 0fa0 0018 97c1 0001 0203 0405  
 0x0030: 0607 0809 0a0b 0c0d 0e0f
```

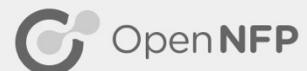
Packet Returned:

```
15:00:43.044892 IP 10.0.0.1.3000 > 10.0.0.100.4000:  
 0x0000: ffff ffff ffff ffff ffff 0800 4500  
 0x0010: 002c 0001 0000 4011 665c 0a00 0001 0a00  
 0x0020: 0064 0bb8 0fa0 cafe babe dead beef 0018  
 0x0030: 97c1 0001 0203 0405 0607 0809 0a0b 0c0d  
 0x0040: 0e0f 0000
```

```
action addHeader() {  
    hdr.custom_header.custom_protocol = 0xCAFEBABE;  
    hdr.custom_header.custom_field = 0xDEADBEEF;  
    hdr.custom_header.setValid();  
}
```

```
control Deparser(in Headers hdrs, packet_out packet)  
{  
    apply {  
        packet.emit(hdrs.ethernet);  
        packet.emit(hdrs.ipv4);  
        packet.emit(hdrs.udp);  
        packet.emit(hdrs.custom_header);  
    }  
}
```

- ▶ Open-NFP.org
 - Worldwide community-driven organization.
 - Collaborative research in the area of network function processing in server networking hardware.
 - Academic and data center networking communities to conduct cutting-edge research and development in the areas of server-based networking datapath offload and acceleration techniques.
- ▶ github.com/open-nfpsw
- ▶ github.com/vmware/p4c-xdp
- ▶ netronome.com/documents/143/UG_Getting_Started_with_eBPF_Offload.pdf
- ▶ p4.org



A photograph of a modern interior space featuring a curved staircase with glass railings. A person is walking down the stairs, their silhouette visible against the bright light coming from a large window on the left. The architecture is clean and contemporary.

NETRONOME

Thank You