Multi-Tenant Programmable Data Planes

Radostin Stoyanov

PhD Candidate, University of Oxford & Software Engineer, Red Hat

Supervisor: Prof. Noa Zilberman

DevConf.CZ - January 28-29, 2022

Outline of Talk

- Data Plane Programmability
- P4 Programming Language
- Data Plane Virtualization
- MTPSA: Multi-Tenant Programmable Switch
- Live Demo

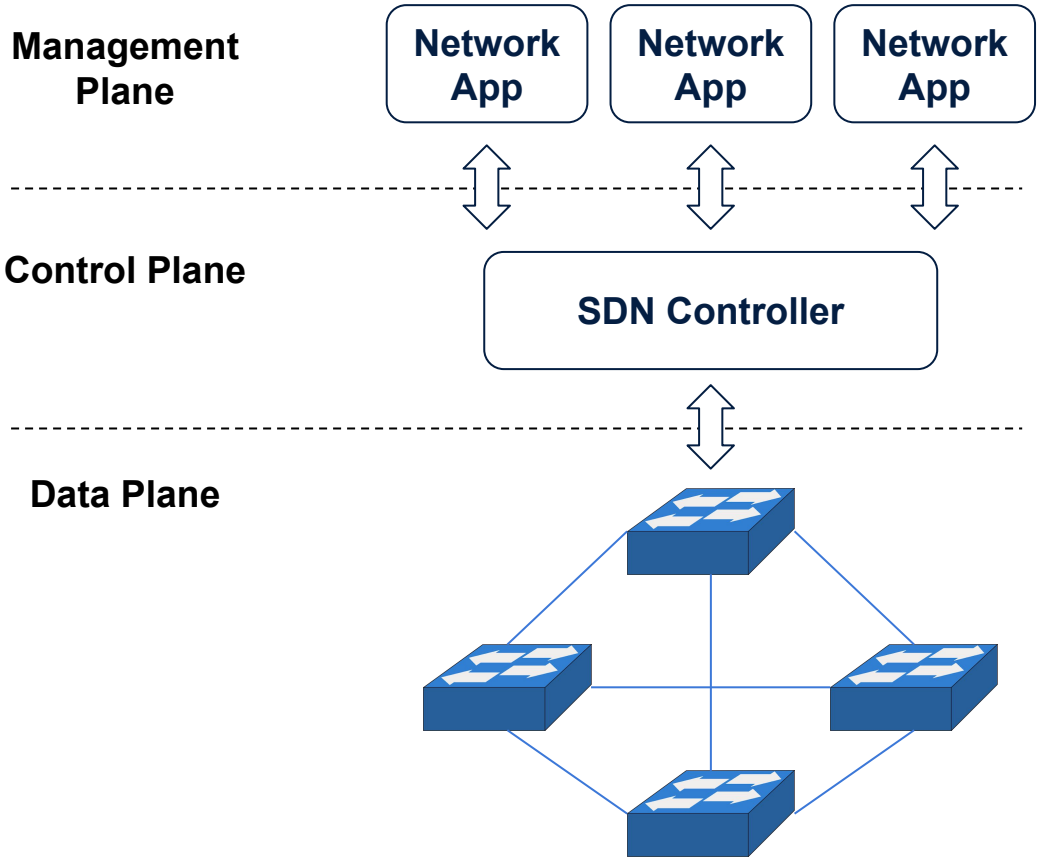
What is a Data Plane?

Software Defined Networking (SDN)

- Centralized Control Plane
- OpenFlow

Drawbacks

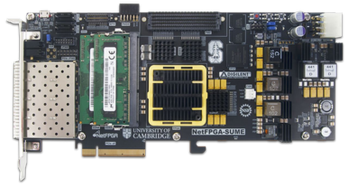
- Protocol evolution requires changes to standards
- Limited interoperability between vendors



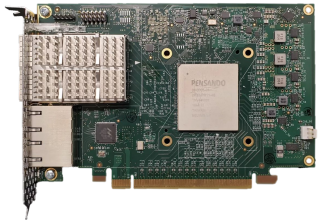
Data Plane Programmability

Hardware

FPGAs



SmartNICs



Switch ASIC

Software

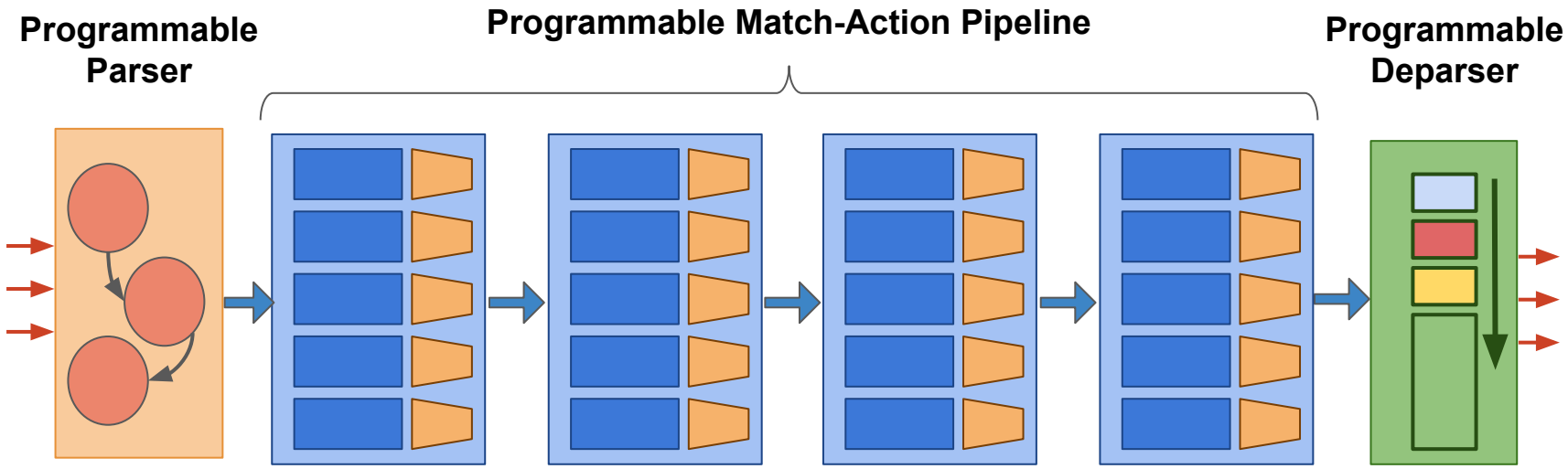
Linux Kernel

eBPF

Kernel-bypass

DPDK
DATA PLANE DEVELOPMENT KIT

PISA: Protocol Independent Architecture



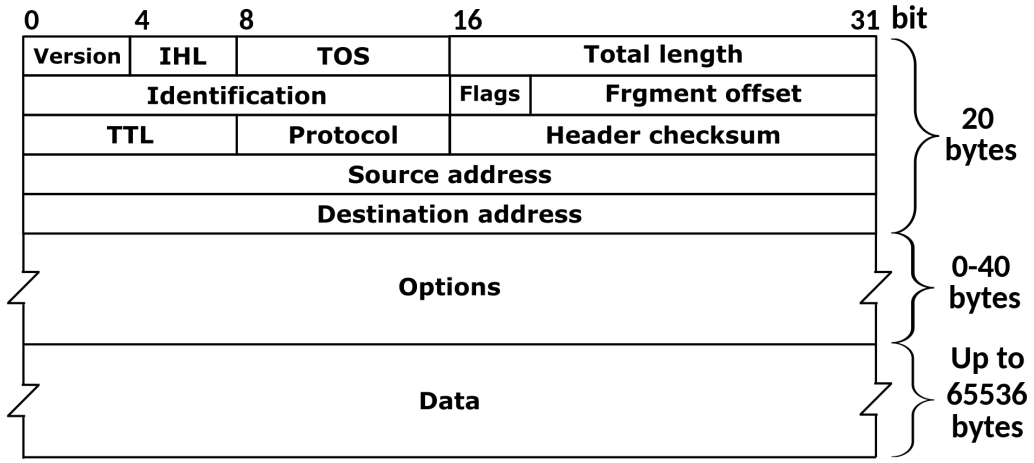
Programming Language

P4 Programming Language

Packet Headers

```
header ipv4_t {
    bit<4>    version;
    bit<4>    ihl;
    bit<8>    diffserv;
    bit<16>   totalLen;
    bit<16>   identification;
    bit<3>    flags;
    bit<13>   fragOffset;
    bit<8>    ttl;
    bit<8>    protocol;
    bit<16>   hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}
```

Packet Structure

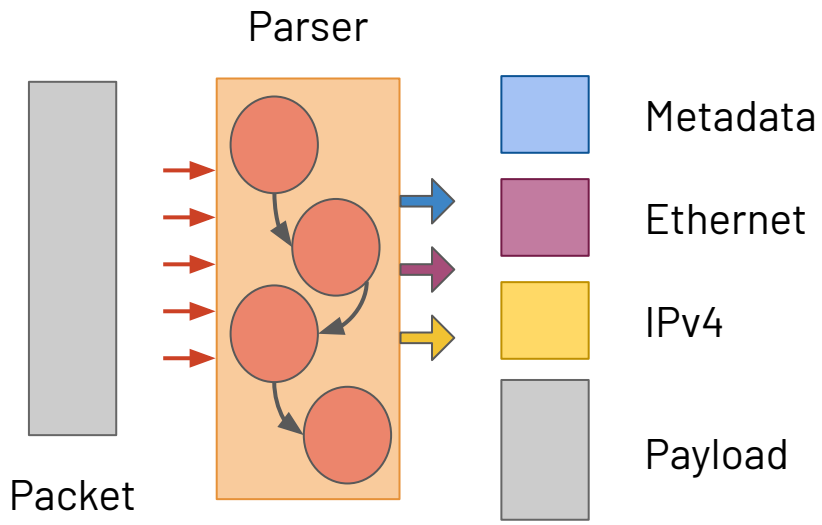


https://en.wikipedia.org/wiki/IPv4

P4 Programming Language

Packet Parser

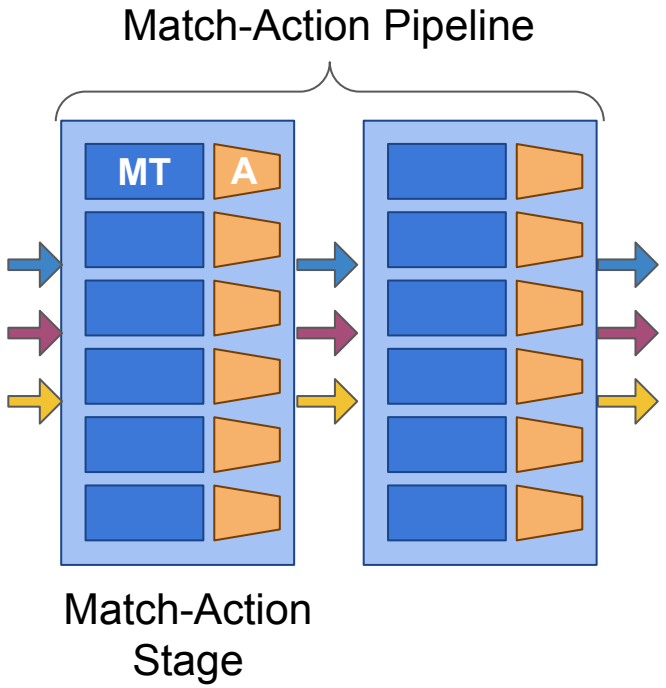
```
state start {
    transition parse_ethernet;
}

state parse_ethernet {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.etherType) {
        TYPE_IPV4: parse_ipv4;
        default: accept;
    }
}

state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
}
```



P4 Programming Language

Match-Action Tables

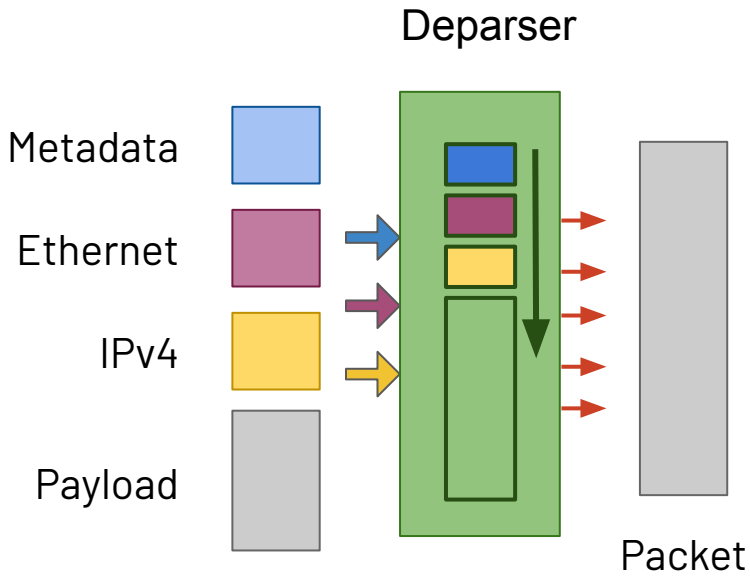
```
table ipv4_lpm {
    key = {
        hdr.ipv4.dstAddr: lpm;
    }
    actions = {
        ipv4_forward;
        drop;
        NoAction;
    }
    size = 1024;
    default_action = drop();
}
```



P4 Programming Language

Deparser

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



Programming Language

```
header ipv4_t {
    bit<4>    version;
    bit<4>    ihl;
    bit<8>    diffserv;
    bit<16>   totalLen;
    bit<16>   identification;
    bit<3>    flags;
    bit<13>   fragOffset;
    bit<8>    ttl;
    bit<8>    protocol;
    bit<16>   hdrChecksum;
    ip4Addr_t srcAddr;
    ip4Addr_t dstAddr;
}
```

Headers

```
state start {
    transition parse_ethernet;
}

state parse_ethernet {
    packet.extract(hdr.ethernet);
    transition select(hdr.ethernet.etherType) {
        TYPE_IPV4: parse_ipv4;
        default: accept;
    }
}

state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
}
```

Parser

```
table ipv4_lpm {
    key = {
        hdr.ipv4.dstAddr: lpm;
    }
    actions = {
        ipv4_forward;
        drop;
        NoAction;
    }
    size = 1024;
    default_action = drop();
}
```

Match-Action Tables

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

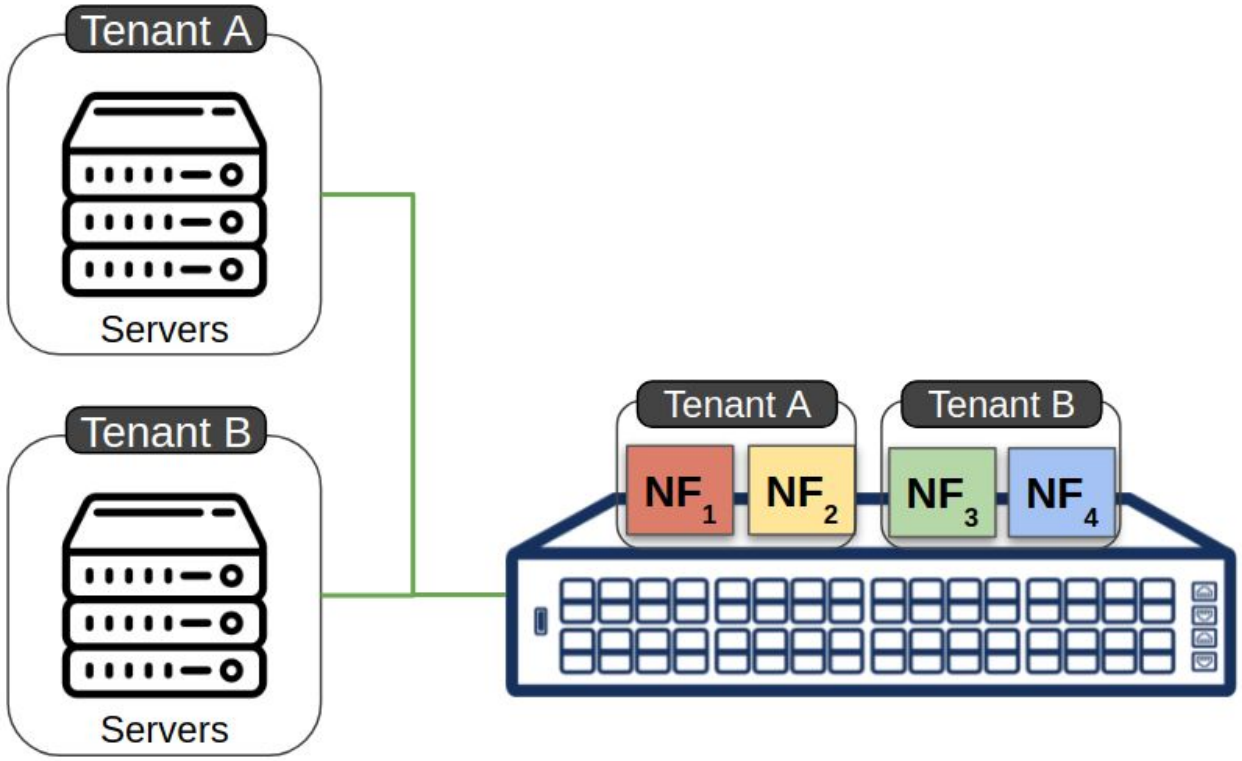
Deparser

Context and Motivation

Context and Motivation

In-network Applications

- In-band Network Telemetry
- Load Balancing
- Machine Learning
- Consensus (Paxos)

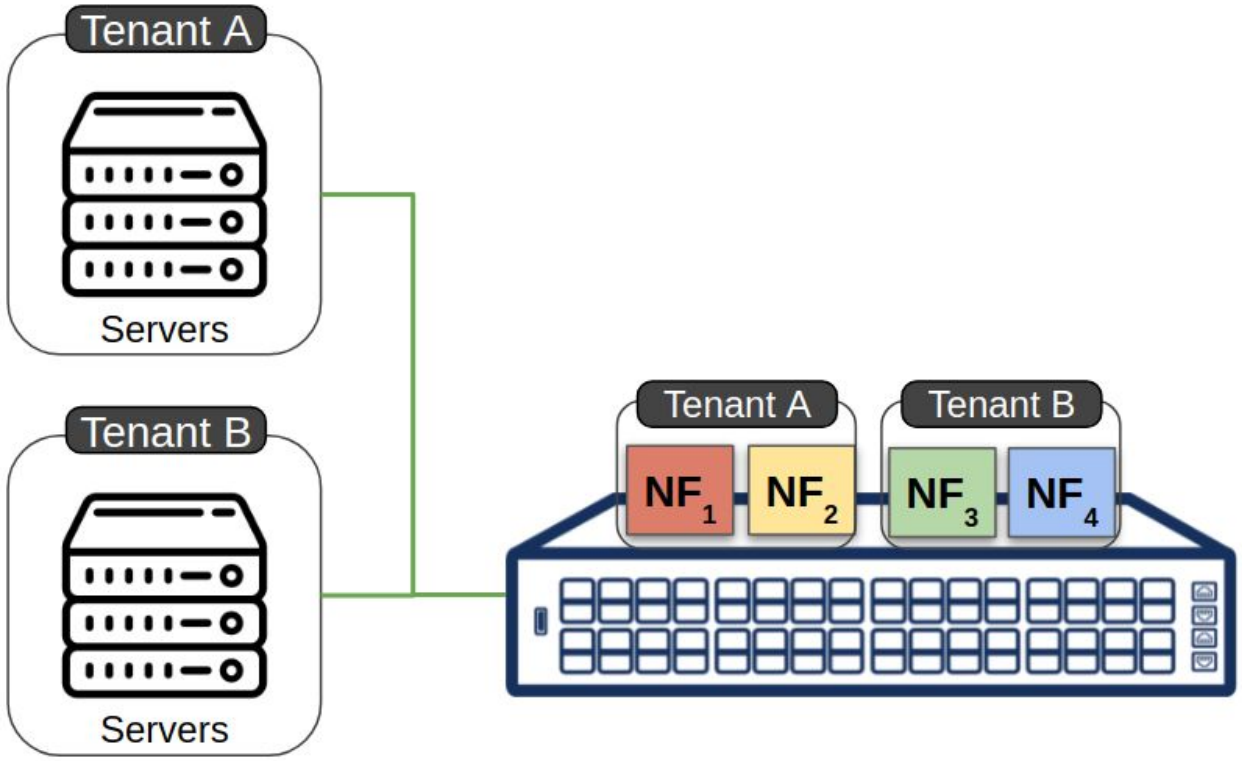


Context and Motivation

Can we support multi-tenancy?

Requirements:

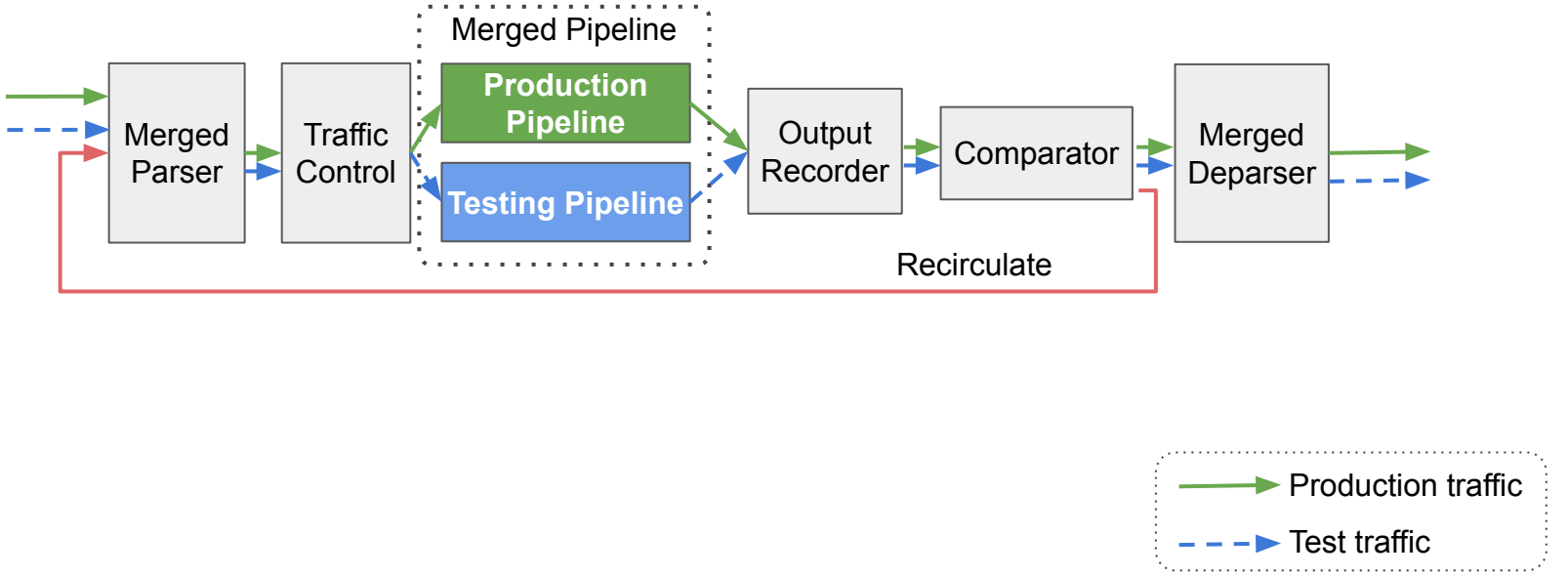
- Security isolation
- Performance isolation
- Resource isolation
- Runtime reconfigurability



Data Plane Virtualization

Data Plane Virtualization

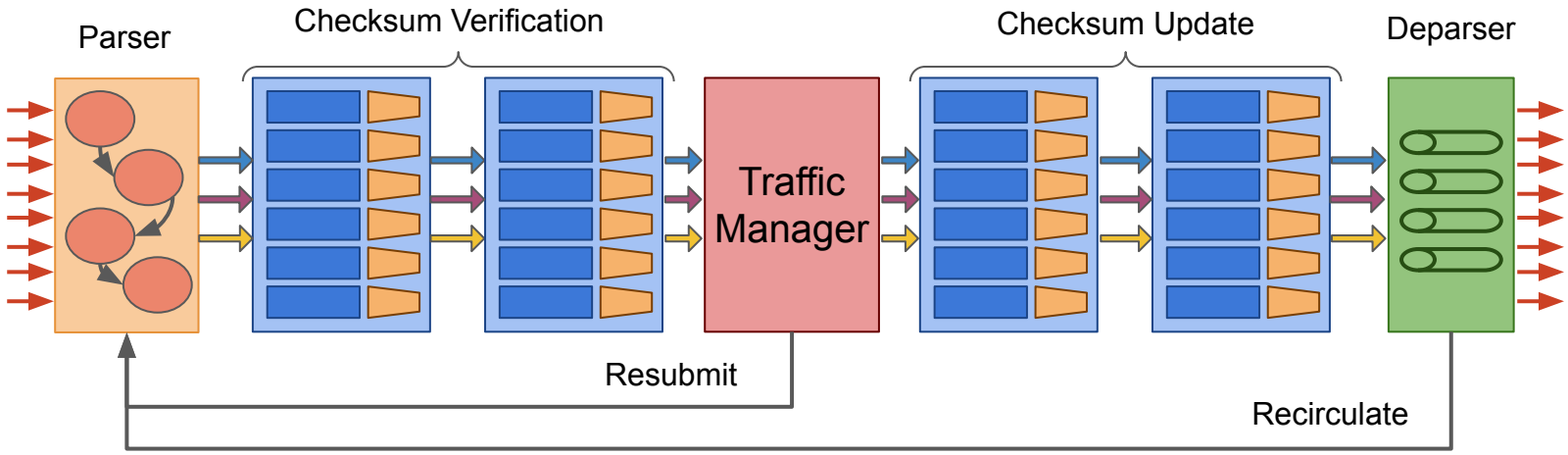
Compiler-based Virtualization



"P4Visor: Lightweight Virtualization and Composition Primitives for Building and Testing Modular Programs", Zheng P, Benson T, Hu C., CoNEXT '18

Data Plane Virtualization

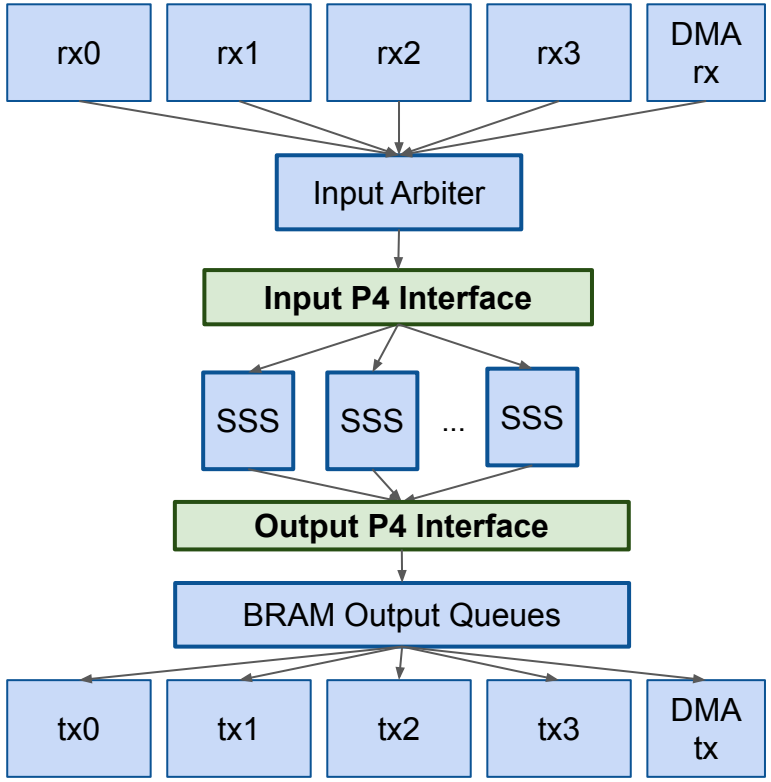
Recirculation-based Virtualization



"HyPer4: Using P4 to Virtualize the Programmable Data Plane", Hancock, David, and Jacobus Van der Merwe, CoNEXT '16

Data Plane Virtualization

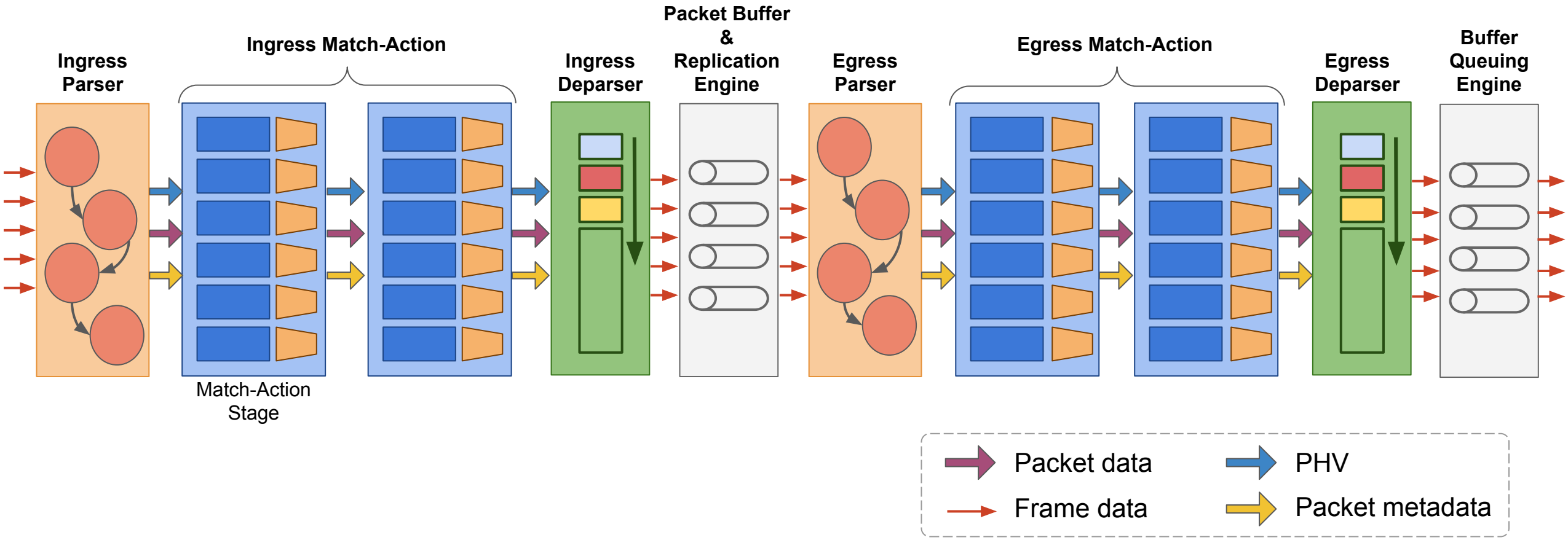
Architecture-driven Virtualization



"P4VBox: Enabling P4-Based Switch Virtualization", Saquetti M, Bueno G, Cordeiro W, Azambuja JR, 2020

Multi-Tenant Programmable Switch

Portable Switch Architecture (PSA)



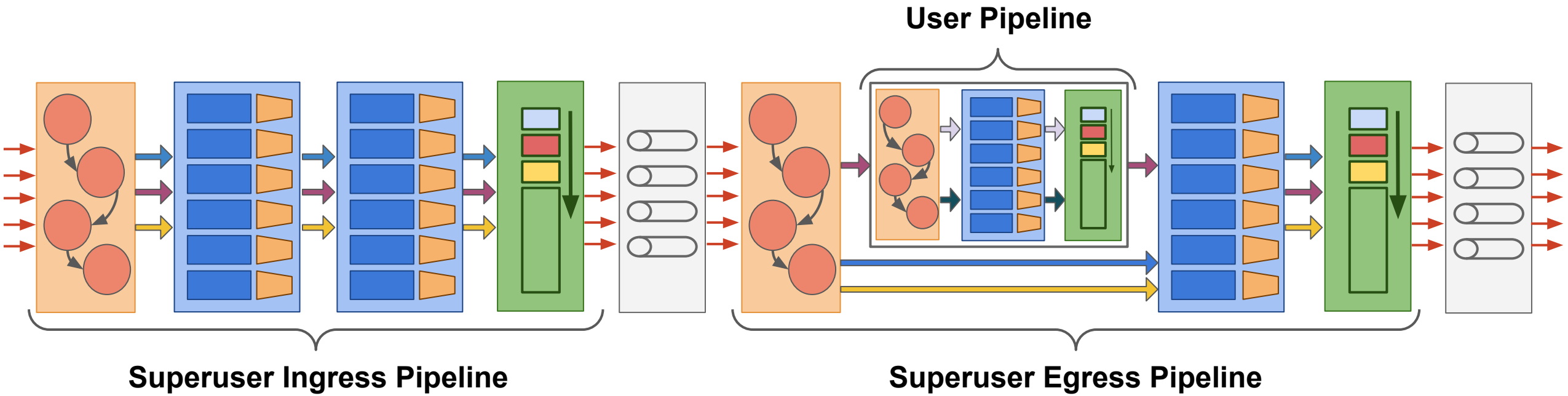
https://p4.org/specs

Multi-Tenant Programmable Switch

- User P4 programs can be compiled and loaded as separate modules
- Each user P4 program runs in a separate context
- Each packet is associated with a P4 program in the pipeline
- Packet processing before and after *user* programs

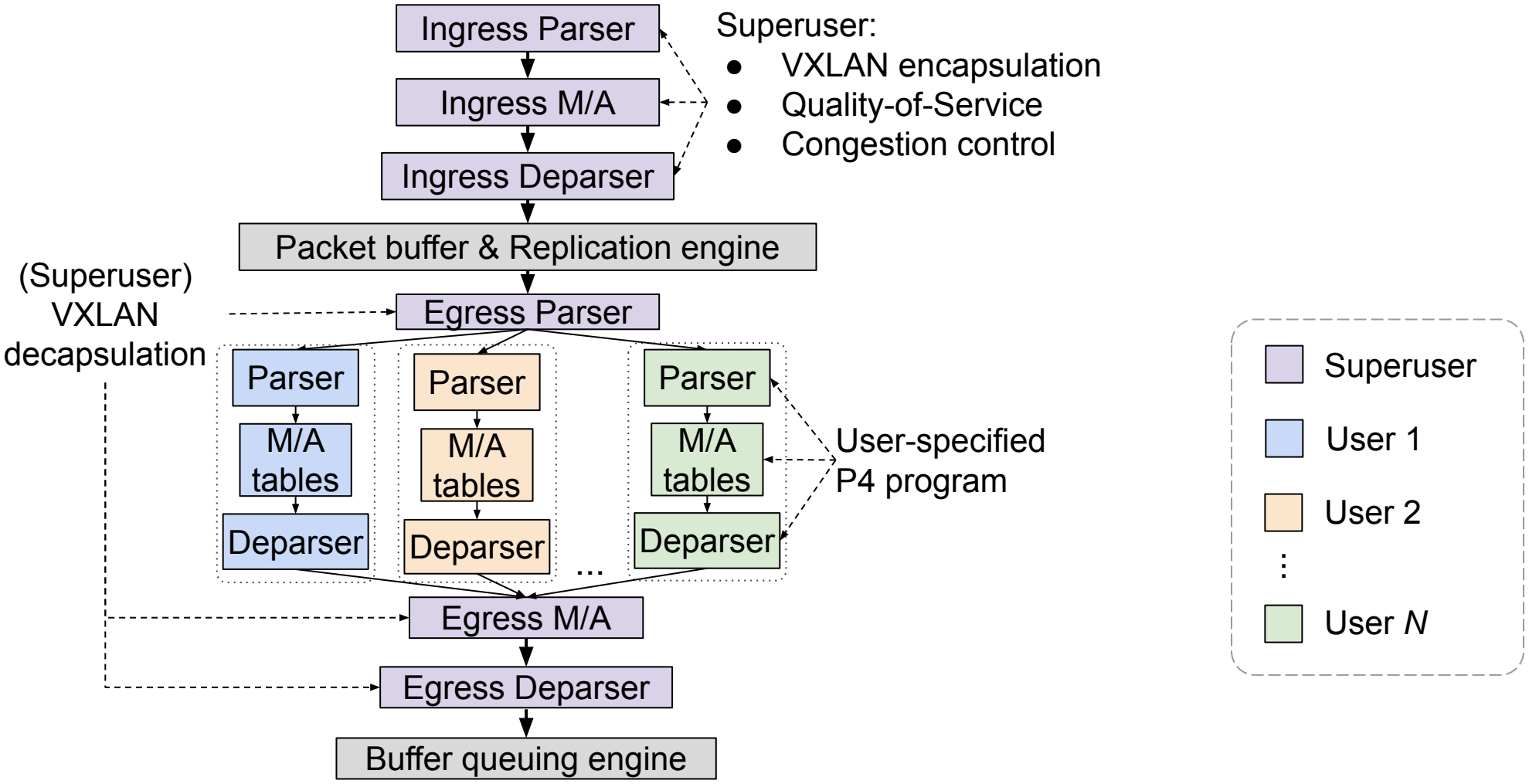
"MTPSA: Multi-Tenant Portable Switch Architecture", Radostin Stoyanov, and Noa Zilberman, EuroP4. 2020.

Multi-Tenant Programmable Switch



"MTPSA: Multi-Tenant Portable Switch Architecture", Radostin Stoyanov, and Noa Zilberman, EuroP4. 2020.

MTPSA Packet Flow



MTPSA User Permissions

Superuser Ingress

Defines permissions given to users to perform certain operations

```
action set_user_id(bit<16> user_id) {
    ostd.user_id = user_id;

    if (user_id == 3) {
        // Disable counter extern in User 3
        ostd.user_permissions = 0x01;
    }
}

table set_user_id_table {
    key = {
        istd.ingress_port: exact;
    }
    actions = {
        set_user_id;
        drop;
    }
    size = 1024;
    default_action = drop();
}
```

MTPSA User Permissions

Superuser Ingress

Defines permissions given to users to perform certain operations

```
action set_user_id(bit<16> user_id) {
    ostd.user_id = user_id;

    if (user_id == 3) {
        // Disable counter extern in User 3
        ostd.user_permissions = 0x01;
    }
}

table set_user_id_table {
    key = {
        istd.ingress_port: exact;
    }
    actions = {
        set_user_id;
        drop;
    }
    size = 1024;
    default_action = drop();
}
```

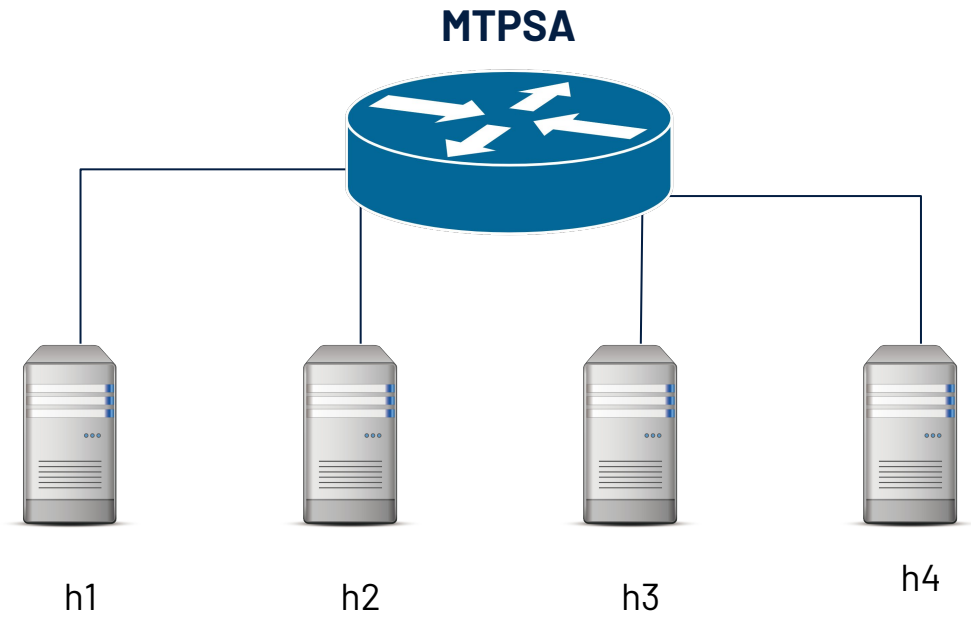
```
$ mtpsa_switch_CLI

RuntimeCmd: switch_context 1
Obtaining JSON from switch...
Done
RuntimeCmd: counter_read ingress.port_data 0
ingress.port_data[0]= (4042 bytes, 28 packets)


RuntimeCmd: switch_context 3
Obtaining JSON from switch...
Done
RuntimeCmd: counter_read ingress.port_data 0
ingress.port_data[0]= (0 bytes, 0 packets)
```

Users can read only their own counters

User 3 does not have sufficient permissions to use counters

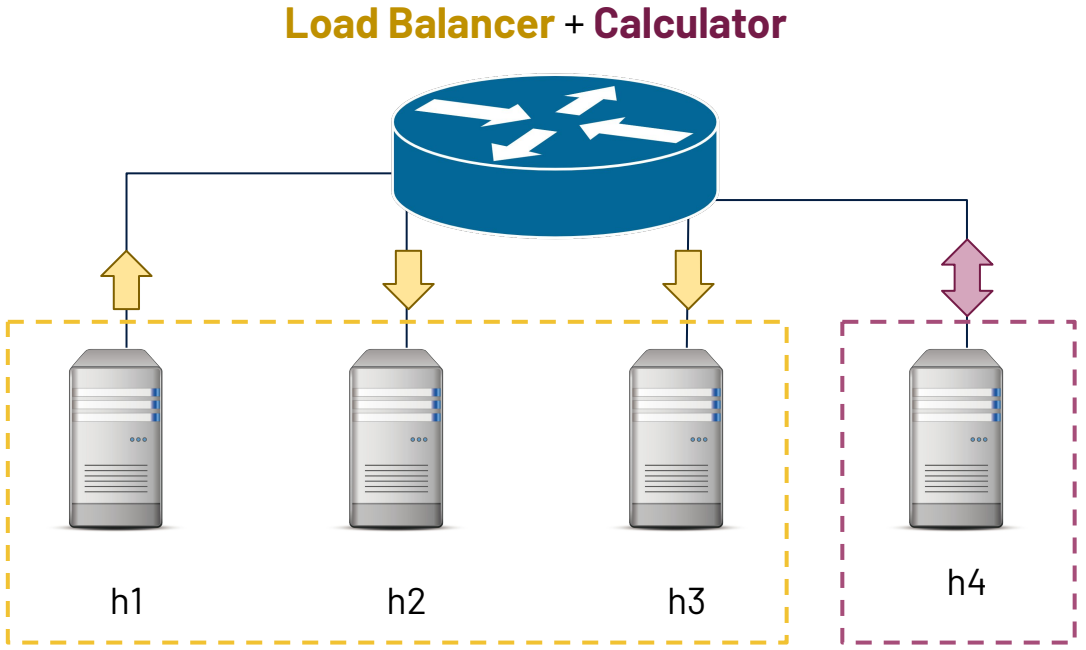
Live Demo

Topology: 4x Hosts + 1x Switch



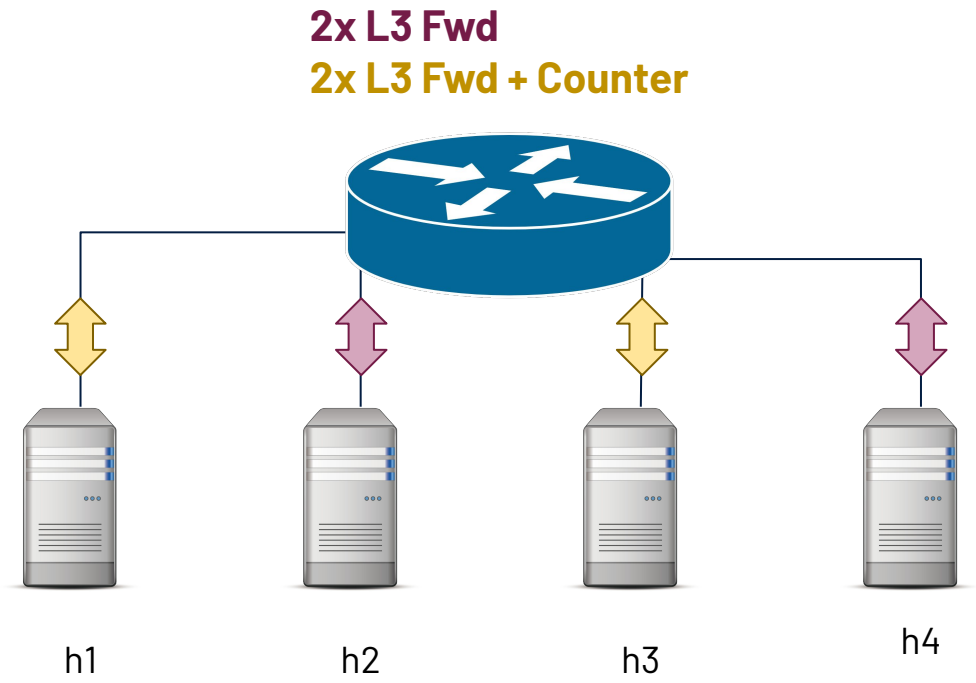
Live Demo

Topology: 4x Hosts + 1x Switch



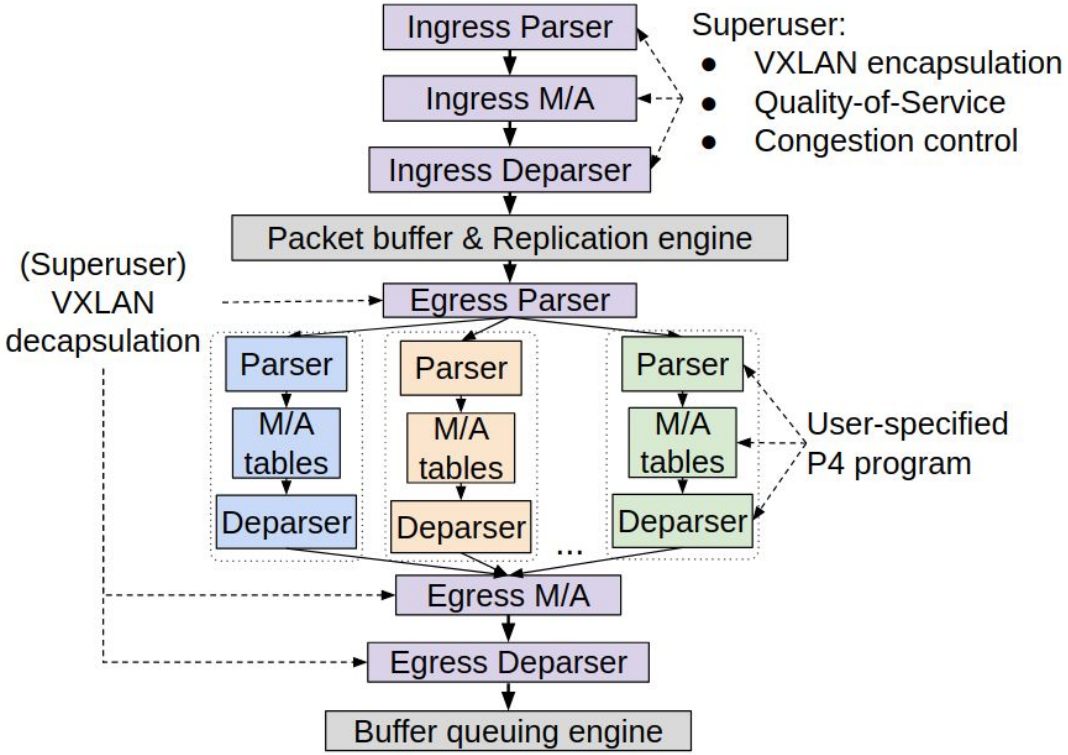
Live Demo

Topology: 4x Hosts + 1x Switch



Takeaways

- User P4 programs can be compiled and loaded as separate modules
- Each user P4 program runs in a separate context
- Each packet is associated with a P4 program in the pipeline
- Packet processing before and after *user* programs



Summary & Questions?

Multi-Tenant Programmable Data Planes

- **Superuser Pipeline**
- **User Pipelines**
- **User Permissions**

https://github.com/mtpsa

https://eng.ox.ac.uk/computing/projects/in-network-computing/mtpsa