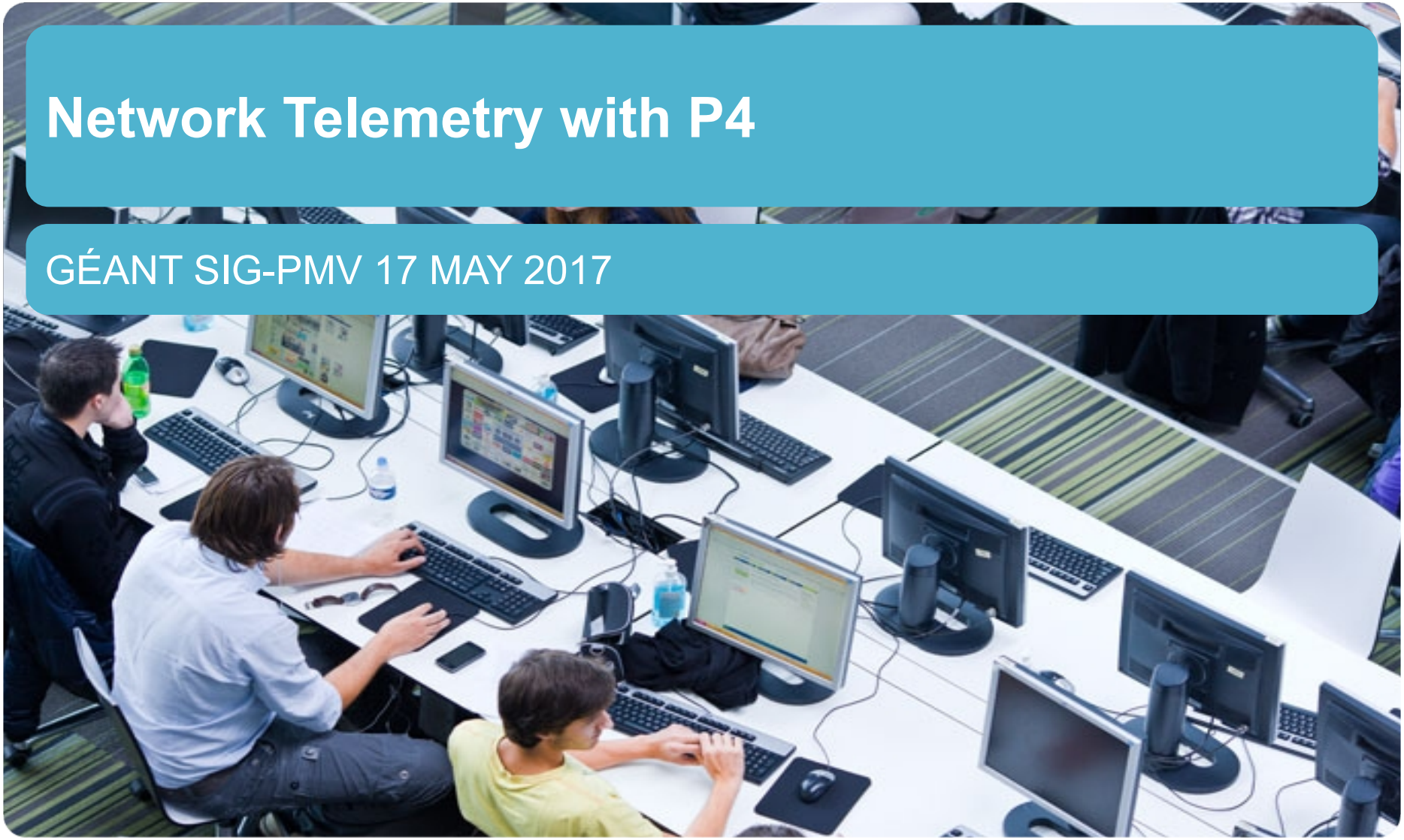
Network Telemetry with P4

GÉANT SIG-PMV 17 MAY 2017



Ronald van der Pol <Ronald.vanderPol@SURFnet.nl>

SURF NET

P4 Language

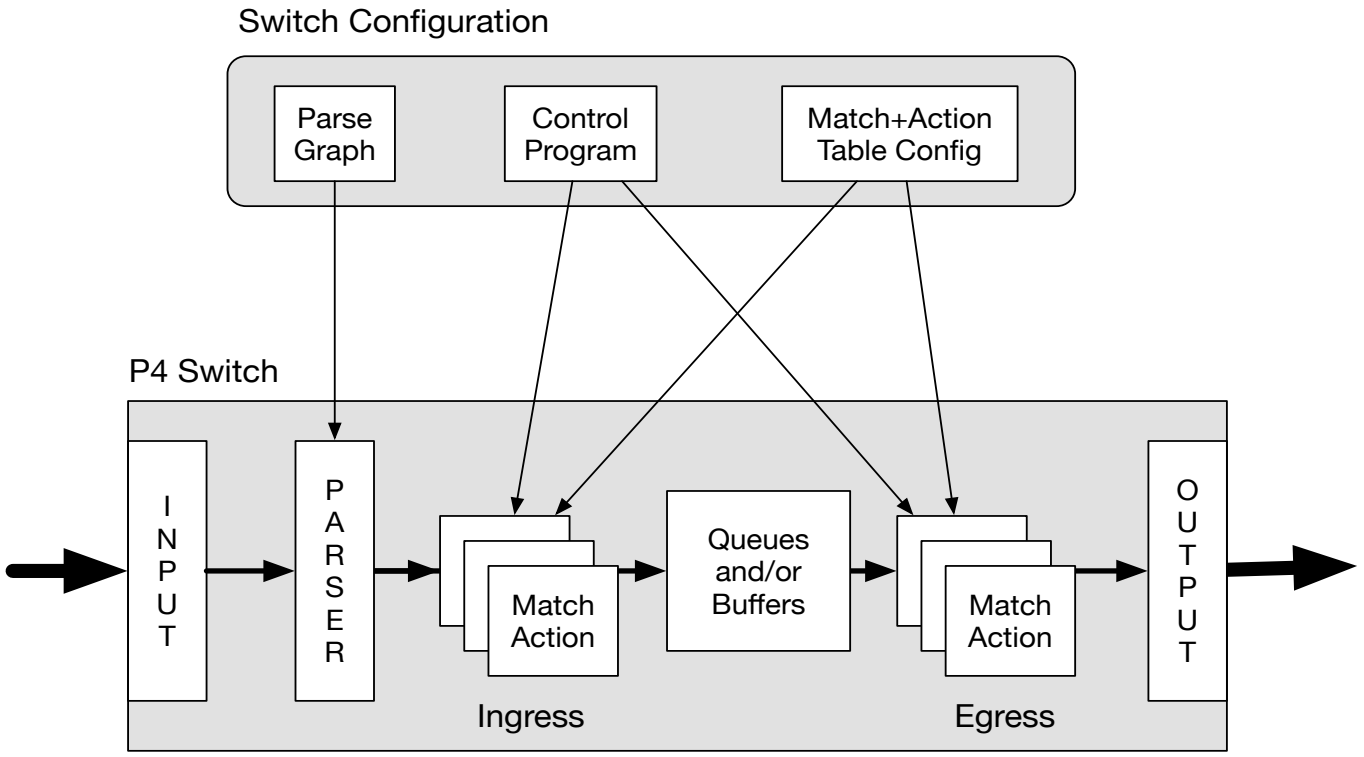
- P4: Programming Protocol-Independent Packet Processors
- Domain Specific Language for network protocols
- Takes the OpenFlow match/action concepts much further
- P4 program defines headers, parser and lookup tables
- P4 program → P4 compiler → target code → P4 switch/NIC
- Target code is loaded on P4 hardware/software switch/NIC

P4 Language Consortium (P4.org)

- Established in 2015, with objectives:

- Build a thriving open source community dedicated to the use and improvement of the P4 language
- Utilize P4 to describe how a forwarding plane should process packets
- Promote standardization and improvement of the P4 language
- Enable industry participants to develop new technologies that function in accordance with the specification
- Benefit consumers and the industry by facilitating adaption of the P4 language

P4 Switch



Source: The P4 Language Specification Version 1.0.2

P4 Elements

P4 consists of three elements:

- header definitions (packet headers and metadata)

- lookup tables & actions (match/action)

- packet parser state machine & checksum field lists

P4 Header Definition Examples

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

header ethernet_t ethernet;
header ipv4_t ipv4;
```

```
header_type ipv4_t {
    fields {
        version : 4;
        ihl : 4;
        diffserv : 8;
        totalLen : 16;
        identification : 16;
        flags : 3;
        fragOffset : 13;
        ttl : 8;
        protocol : 8;
        hdrChecksum : 16;
        srcAddr : 32;
        dstAddr: 32;
    }
}
```

P4 Metadata Example (target specific)

Metadata is associated with each packet

```
header_type ingress_metadata_t {
    fields {
        ingress_port: 9;
        packet_length: 14;
        egress_spec: 16;
        instance_type: 4;  /* normal, clone, recirculated */
    }
}

header ingress_metadata ingress_metadata;
```

P4 Parser

```
parser start {
    return parse_ethernet;
}

parser parse_ethernet {
    extract(ethernet);
    return select(latest.etherType) {
        ETHERTYPE_IPV4 : parse_ipv4;
        ETHERTYPE_IPV6: parse_ipv6;
        default: ingress;
    }
}

parser parse_ipv4 {
    return select(lastest.<some IPv4 field>) {
        <some field value>: do_something;
        default ingress;
}
```

P4 L2 Switch Table Example

```
table dmac {
    reads {
        ethernet.dstAddr: exact;
    }
    actions {
        forward;
        broadcast;
    }
    size : 16384;
}
```

P4 L2 Switch Forward Action

```
action forward(port) {
        modify_field(standard_metadata.egress_port, port);
}
```

Counters, Meters & Registers

```
counter direct_counter {
        type: bytes;
        direct: dmac;
}
```

```
register R0 {
        width: 64;
        instance_count: 1;
}

action update_reg {
        add_to_field(meta.value, 42);
        register_write(R0, 0, meta.value);
}
```

```
counter indirect_counter {
        type: packets;
        instance_count: 16384;
}

action foobar(idx) {
        count(indirect_counter, idx);
        forward;
}
```

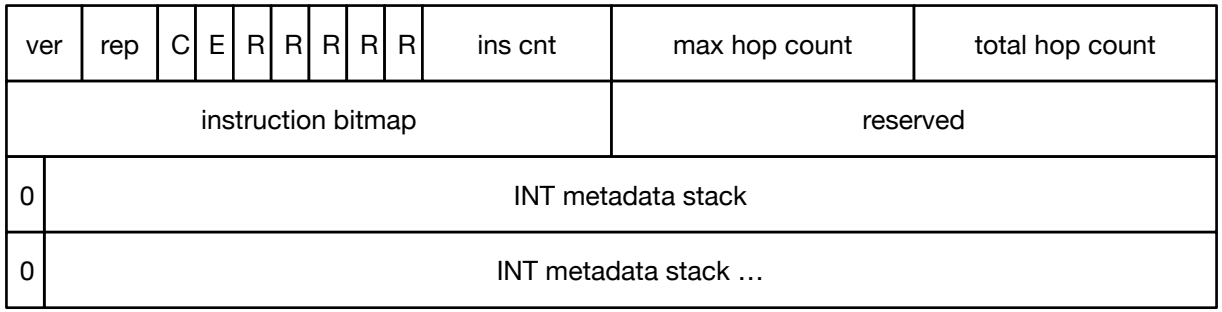
Primitive Actions

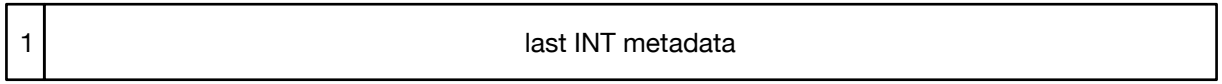
modify_field(dst, value [, mask])	dst: FLD value: VAL/FLD mask: VAL, identifies bits to change
add_to_field(dst, value)	e.g. add_to_field(ipv4.ttl, -1)
add(dst, val1, val2) **bit_and**(dst, val1, val2) **bit_or**(dst, val1, val2) **bit_xor**(dst, val1, val2) **shift_left**(dst, val1, val2) **shift_right**(dst, val1, val2)	dst: FLD val1: VAL/FLD val2: VAL/FLD
modify_field_with_hash_based_offset (dst, base, field_list_calc, size)	

Inband Telemetry

- INT over VXLAN
- INT over GENEVE
- INT over NSH
- INT over TCP
- INT over UDP (as payload)

INT Header Format





Ver (2b): version (0)
Rep (2b): replication requested
C (1b): copy
E (1b): max hop count exceeded
R (1b): reserved bits
Ins cnt (5b): nr of instructions set
Max hop count (8b): maximum hop count allowed
Total hop count (8b): total nr of hops added

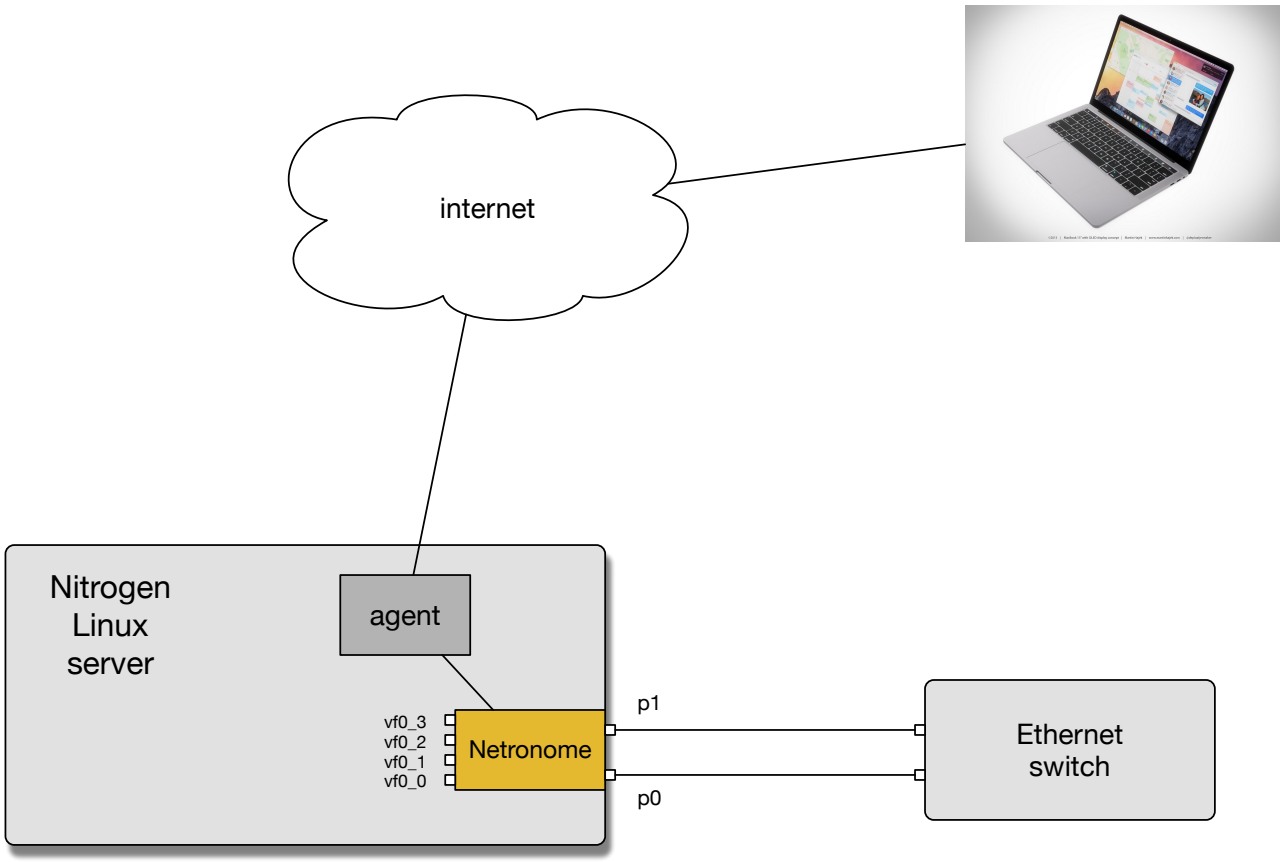
Added by intermediate device

INT Instruction Bitmap

- Bit0: (MSB) switch ID
- Bit1: ingress port ID
- Bit2: hop latency
- Bit3: queue occupancy
- Bit4: ingress timestamp
- Bit5: egress port ID
- Bit6: queue congestion status
- Bit7: egress port Tx utilization

- Rest of bits are reserved

Demo Setup


internet
Nitrogen
Linux
server
agent
vf0_3
vf0_2
vf0_1
vf0_0
Netronome
p1
p0
Ethernet
switch

Workshop

Dataplane Accelaration Developer Day
DXDD Europe 2017
Workshop with hands-on Labs
Wed 7 June 2017
SURFnet, Utrecht

http://www.open-nfp.org/dxdd-europe-2017

Ronal van der Pol
Ronald.vanderPol@SURFnet.nl

WHAT SURF CAN DO