Experiences with Programmable Dataplanes

Ronald van der Pol

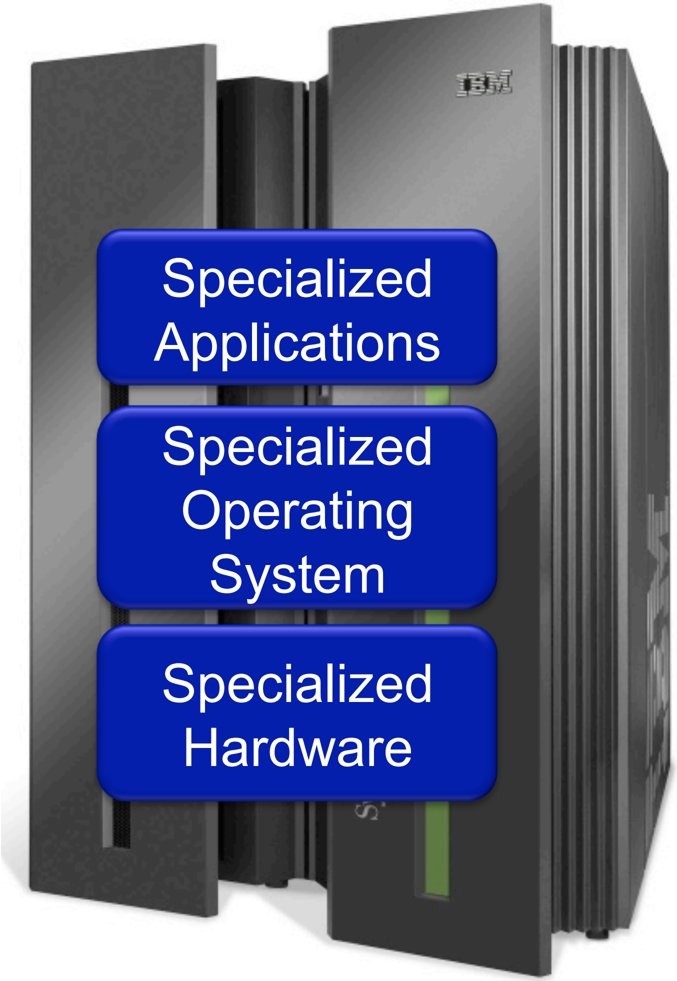
SURFnet

TNC 2016, 12-16 June, Prague (CZ)

Overview

- Motivation for Programmable Dataplanes
- OpenFlow and Pipelines
- Various Network Silicon
- Table Type Patterns (TTPs) and P4
- Summary

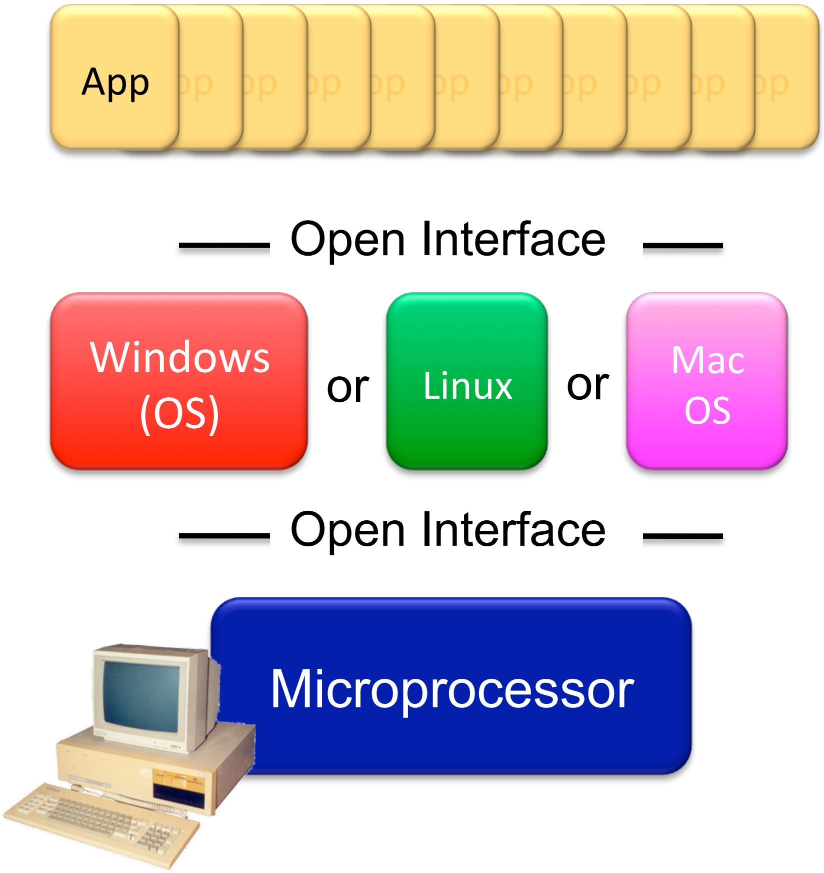
(slide by Nick McKeown, Stanford University)



Specialized Applications

Specialized Operating System

Specialized Hardware



App

Open Interface

Windows (OS) or Linux or Mac OS

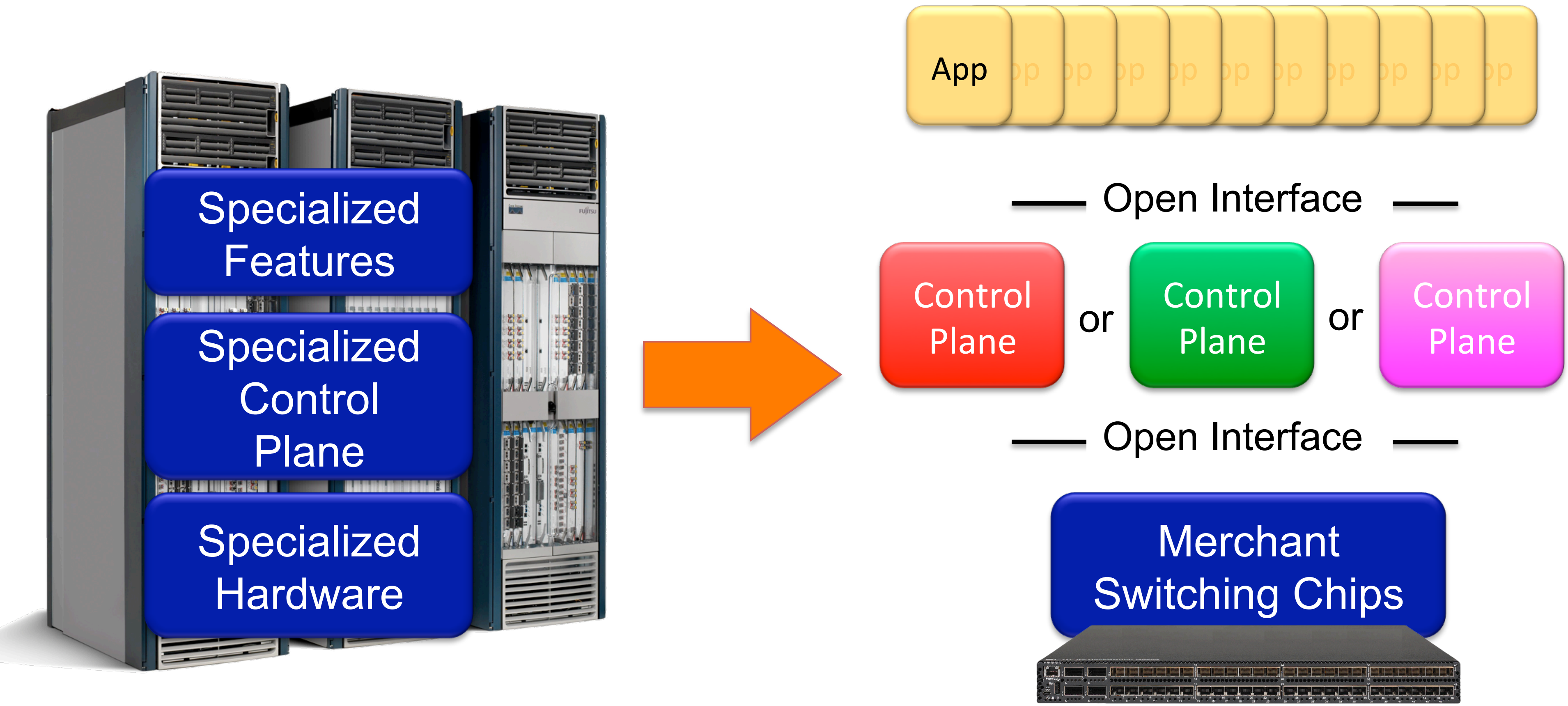
Open Interface

Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

(slide by Nick McKeown, Stanford University)



Specialized Features

Specialized Control Plane

Specialized Hardware

App

Open Interface

Control Plane or Control Plane or Control Plane

Open Interface

Merchant Switching Chips

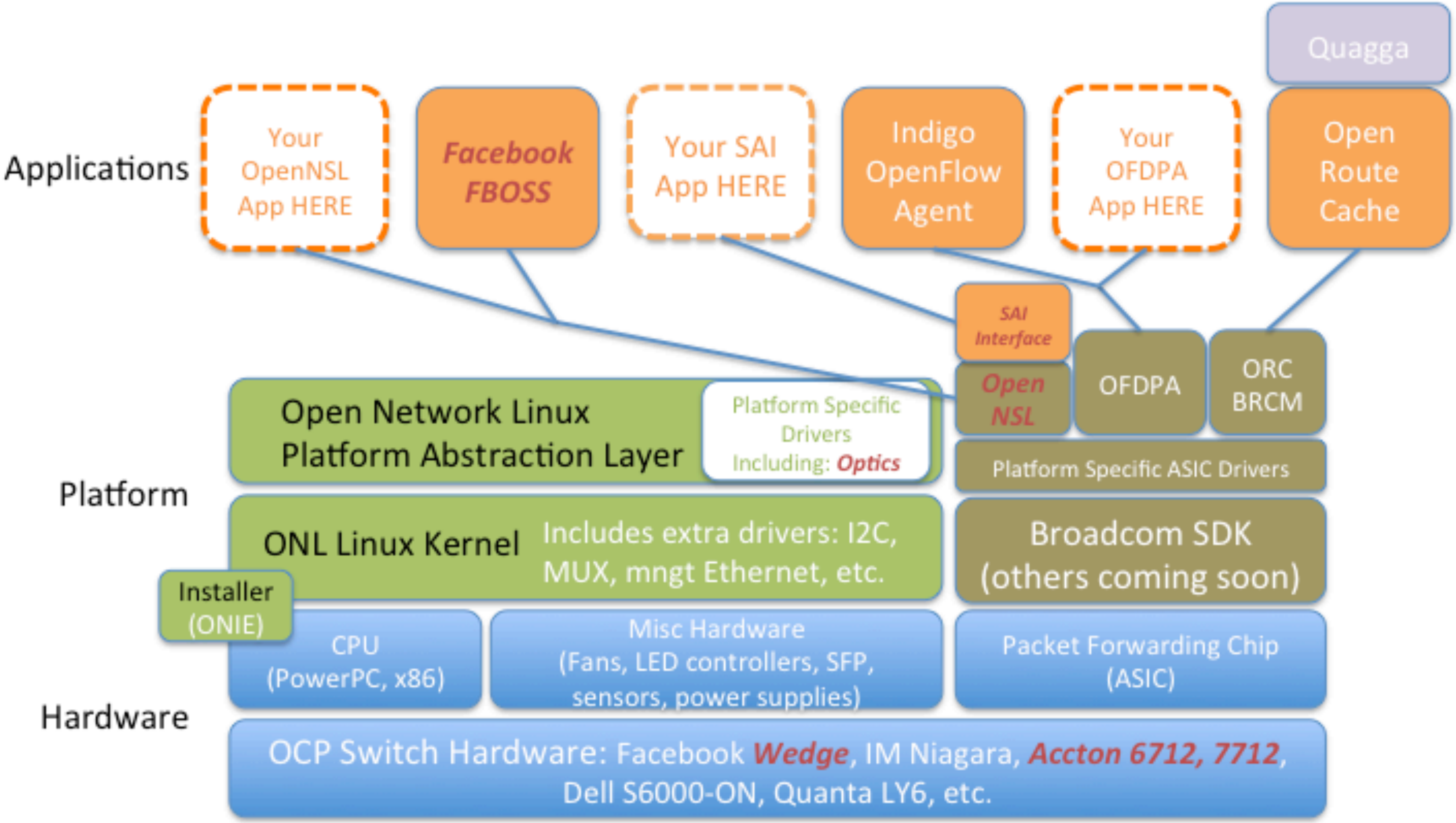
Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

Network Disaggregation

- Best of breed in hardware and software
- Open APIs
- Open Hardware
- User/operator in control
  - Not (or less) dependent of vendor roadmaps
  - Implement and experiment with new protocols

Network Disaggregation Ecosystem


Quagga
Applications
Your OpenNSL App HERE
Facebook FBOSS
Your SAI App HERE
Indigo OpenFlow Agent
Your OFDPA App HERE
Open Route Cache
SAI Interface
Open NSL
OFDPA
ORC BRCM
Open Network Linux Platform Abstraction Layer
Platform Specific Drivers Including: Optics
Platform Specific ASIC Drivers
Platform
ONL Linux Kernel
Includes extra drivers: I2C, MUX, mngt Ethernet, etc.
Broadcom SDK (others coming soon)
Installer (ONIE)
CPU (PowerPC, x86)
Misc Hardware (Fans, LED controllers, SFP, sensors, power supplies)
Packet Forwarding Chip (ASIC)
Hardware
OCP Switch Hardware: Facebook Wedge, IM Niagara, Accton 6712, 7712, Dell S6000-ON, Quanta LY6, etc.

OpenFlow

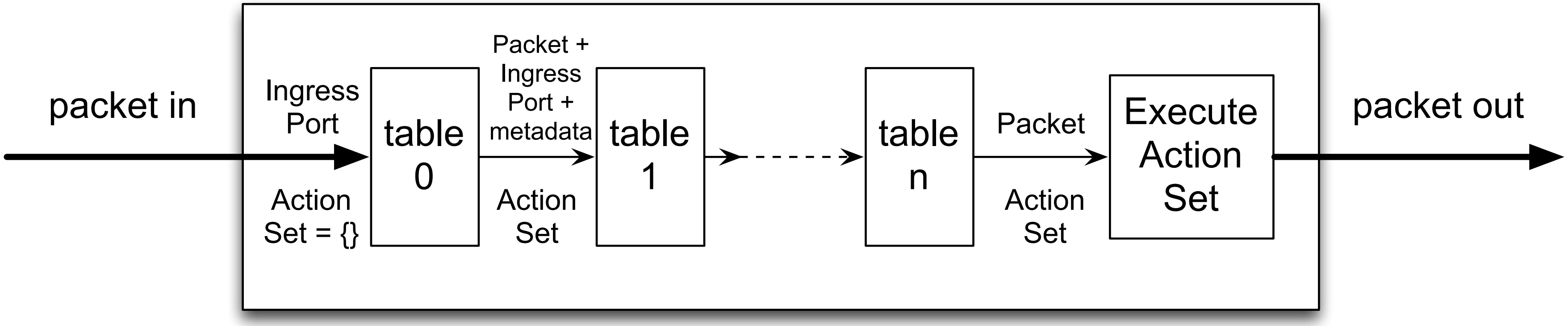
- OpenFlow gives user/operator direct access to flow forwarding tables
- OpenFlow provides Match/Action semantics
- Supported on many hardware switches
  - Pure OpenFlow switches
  - Hybrid switches (conventional switch add-on)
- Many (open source) controller platforms
- OpenFlow started the network disaggregation efforts

SoC ASIC based OpenFlow Switches

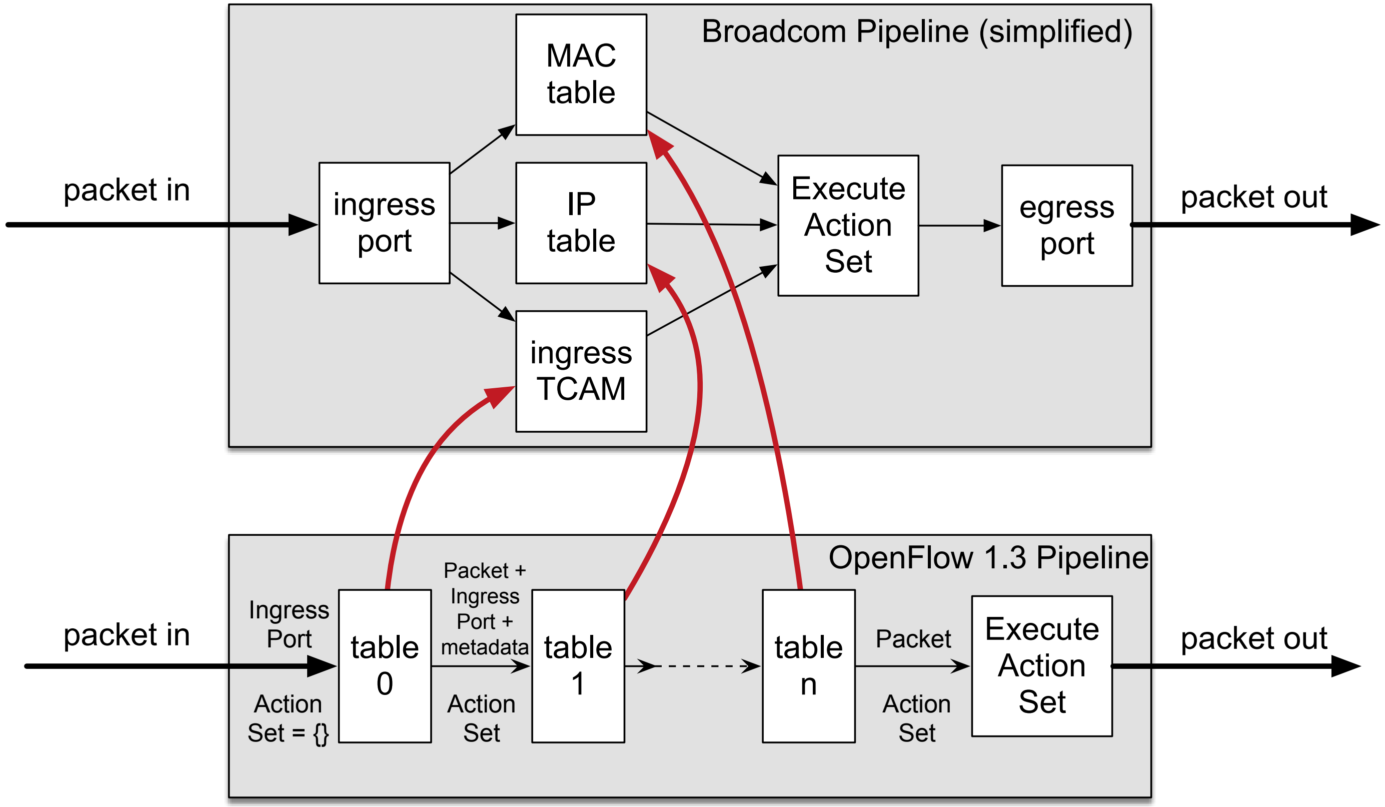
- Many based on Broadcom ASICs (e.g. Trident)
- Only a small fixed amount of lookup tables
  - TCAM (wildcard entries, ACLs)
  - MAC Forwarding Database
  - L3 longest prefix match table
  - L3 host routes

OpenFlow 1.3 Multiple Tables

- Prevent flow entry explosion
- Multi-table pipeline



Mapping of Flow Tables



based on Pica8 documentation

Broadcom Trident II

- There is very little public technical information because of Broadcom's NDA
- Several TCAMs, L2, L3, LPM tables
- Unified Forwarding Table (UFT) memory banks can be allocated to:
  - L2 entries
  - ARP entries
  - L3 LPM entries
  - Exact match ACL entries

Broadcom Trident II UFT

BANK	SIZE
0	4K x 420 bits
1	4K x 420 bits
2	16K x 420 bits
3	16K x 420 bits
4	16K x 420 bits
5	16K x 420 bits
6	1K x 420 bits
7	1K x 420 bits
8	1K x 420 bits
9	1K x 420 bits

Dedicated L2 MAC (32K x 105 bits)

Shared Entries (256K x 105 bits)

Dedicated L3 host entries (16K x 105 bits)

Trident II UFT Combinations

Mode	L2	L3 hosts	LPM
0	288K	16K	0
1	224K	56K	0
2	160K	88K	0
3	96K	120K	0
4	32K	16K	128K (77K – IPv6)

Limitations of SoC ASICs

- Fixed semantics tables (L2, L3, LPM, TCAM)
- Fixed size tables (or limited resizing)
- No recirculation of packets (one pass through pipeline)

ASIC/OpenFlow Limitation Examples

- Limitation of SoC ASICs
  - OpenDaylight Service Function Chaining (SFC) project configures multiple tables
  - These end up in 1 TCAM and does not work
  - Result: generic applications like ODL SFC cannot be used; application needs to be adapted to ASIC
- Limitation of OpenFlow
  - Still dependence on SDOs and vendors for new encapsulations/protocols
  - We want to experiment with Network Services Header (NSH), but no support in OpenFlow

Programmable Network Silicon

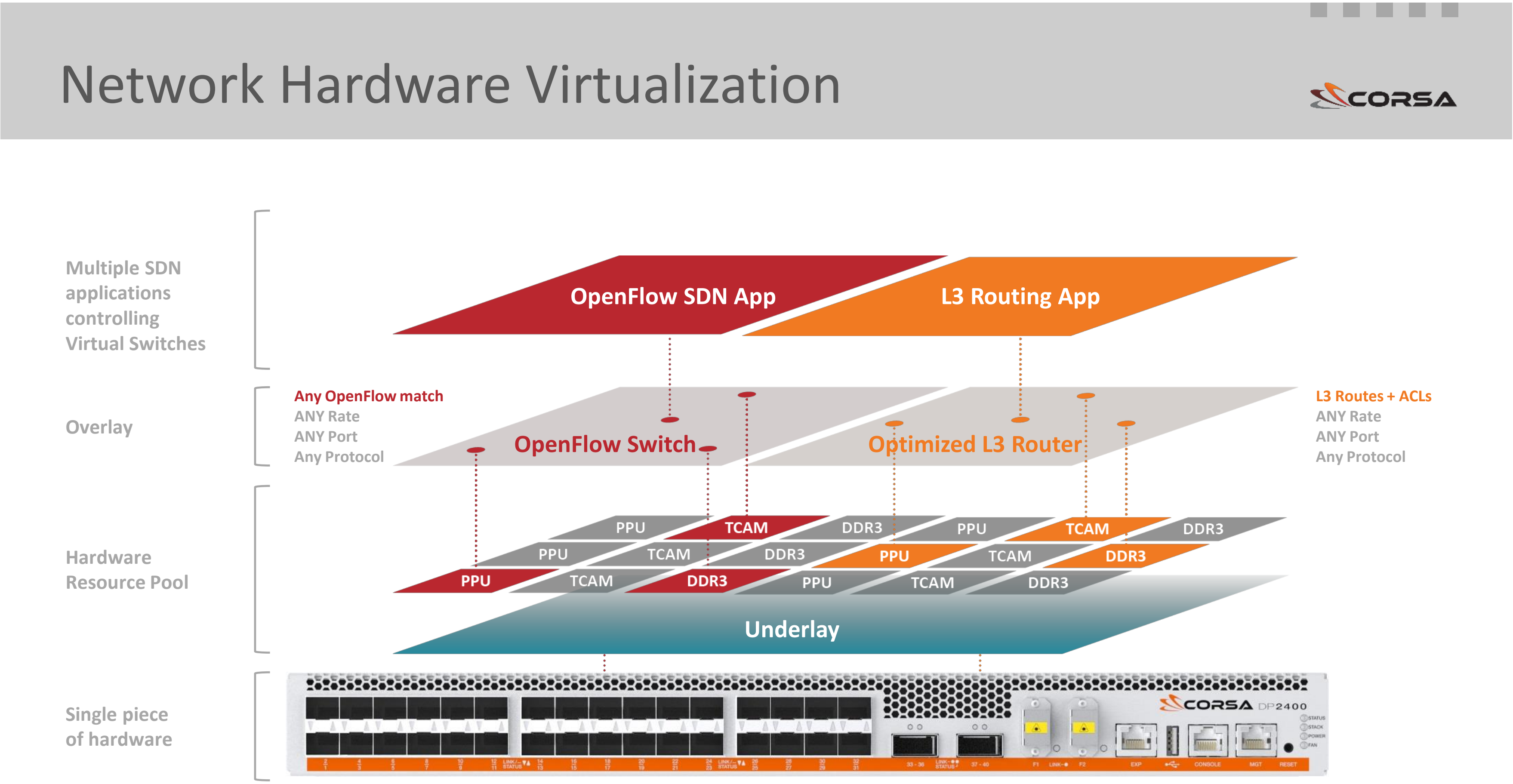
- FPGAs (Field Programmable Gate Arrays) + TCAM + DDR
  - *Corsa DP6410 *)*
- Network Processors (NPUs) + TCAM + DDR
  - *NoviFlow NS2128 *)*
- Flow Processor
  - *Netronome NFP-4000 **)*
- Programmable Switch Silicon
  - *Cavium Xpliant **)*

**) present in SURFnet testbed*

***) soon in SURFnet testbed*

Corsa (FPGA/TCAM/DDR3)

Network Hardware Virtualization



Multiple SDN applications controlling Virtual Switches

OpenFlow SDN App

L3 Routing App

Overlay

Any OpenFlow match
ANY Rate
ANY Port
Any Protocol

OpenFlow Switch

Optimized L3 Router

L3 Routes + ACLs
ANY Rate
ANY Port
Any Protocol

Hardware Resource Pool

PPU TCAM DDR3 PPU TCAM DDR3

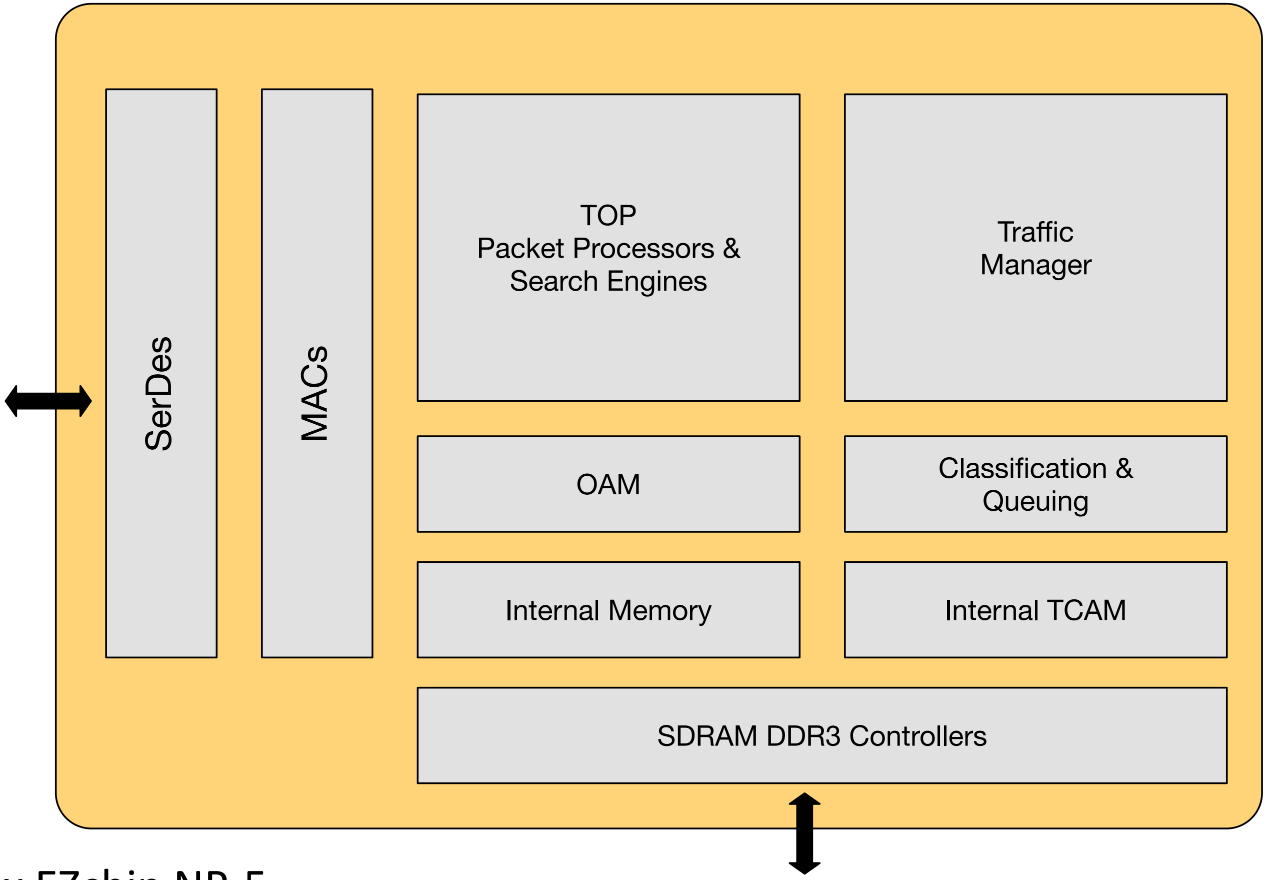
PPU TCAM DDR3 PPU TCAM DDR3

PPU TCAM DDR3 PPU TCAM DDR3

Underlay

Single piece of hardware

NoviFlow NS2128



Mellanox EZchip NP-5

NoviFlow Pipeline Configuration

- Set config pipeline <id> <size> <width> <type>
  - <type> is exact (DDR) or wildcard (TCAM)
  - Default
    - 28 wildcard + 28 exact tables
    - 4096 rows
    - 40 byte wide

Pipeline Abstractions

- Flexible programmable pipelines need an abstraction to describe them
- Two popular approaches:
  - Table Type Patterns (TTP) – OpenFlow pipelines
  - P4 (Programming Protocol-Independent Packet Processors)
- Both can be used to
  - Let the switch advertise its supported pipeline(s)
  - Tell the switch what pipeline to construct

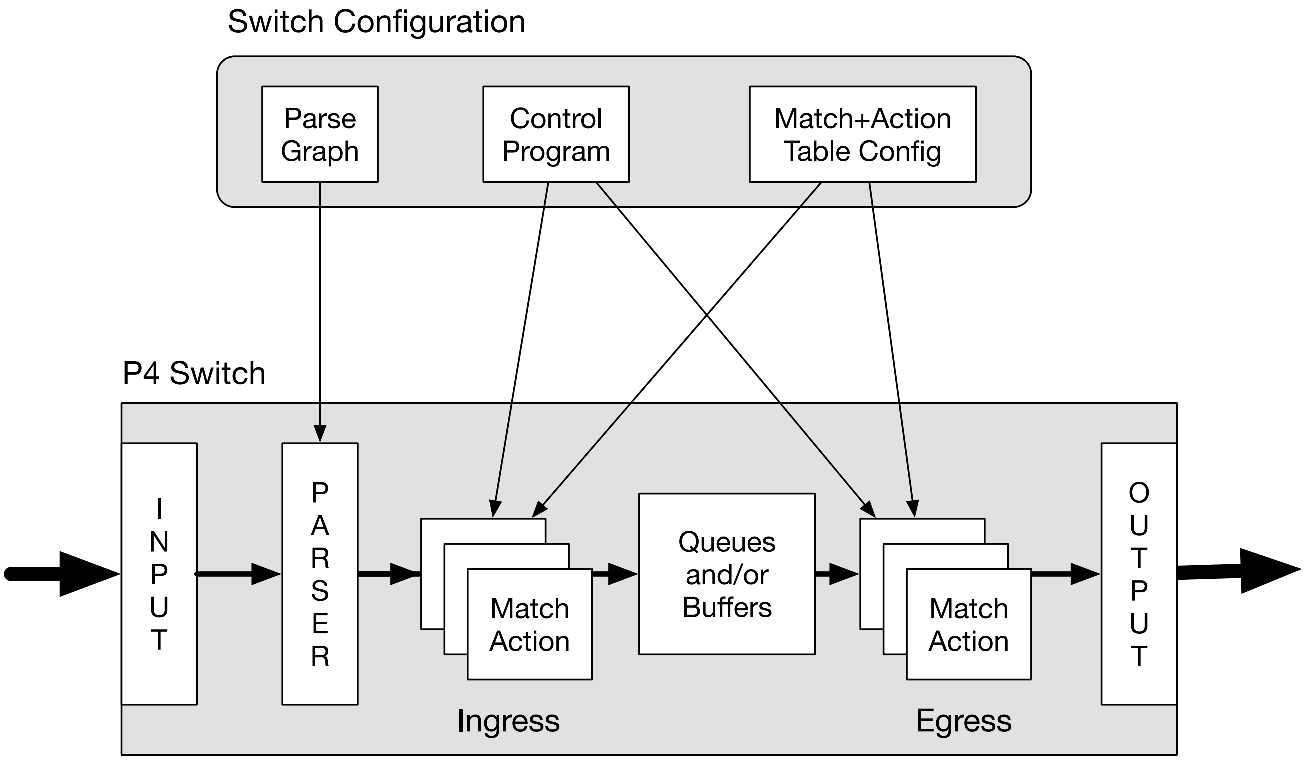
Table Type Patterns (TTPs)

- A TTP is an abstract model that describes (in JSON syntax) the forwarding behaviour
  - Description of flow tables
  - Description of valid flow_mods, group_mods and meter_mods
- Switch and controller may support multiple TTPs
- At startup there is a negotiation between switch and controller about which TTP to use

P4 Language

- P4: Programming Protocol-Independent Packet Processors
- Domain Specific Language for programmable dataplanes
- P4 program → P4 compiler → target code
- Target code is loaded on P4 switch
    - Consists of packet parser and lookup tables

P4 Switch



Source: The P4 Language Specification
Version 1.0.2

Example P4 Header Definitions

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

```
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;
    }
}
```

Example P4 Parser

```
parser start {
    return parse_ethernet;
}

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

parser parse_ipv4 {
    extract(ipv4);
    return ingress;
}
```

P4 Supported Table Types

- ***Exact***: value == table entry
  - E.g. IPv4 host route
- ***Ternary***: value AND mask == table entry
  - Wildcard
- ***LPM***: Longest Prefix Match
  - Special case of ternary (1111….11110000…..0000)
- ***Range***: low entry <= value <= high entry
- ***Valid***: table entry = {true, false}
  - True: header field is valid
  - False: header field is not valid

P4 Supported Checksum Algorithms

- XOR16
- CSUM16
- CRC16
- CRC32
- Programmable_CRC
  - Arbitrary CRC polynomial

Additional P4 Features

- Counters
  - Type: bytes or packets
  - Min-width
  - Saturating: stop counting; default is wrap
- Meters
- Registers
- Resubmit (original packet + metadata)
- Recirculate (packet after egress modifications)

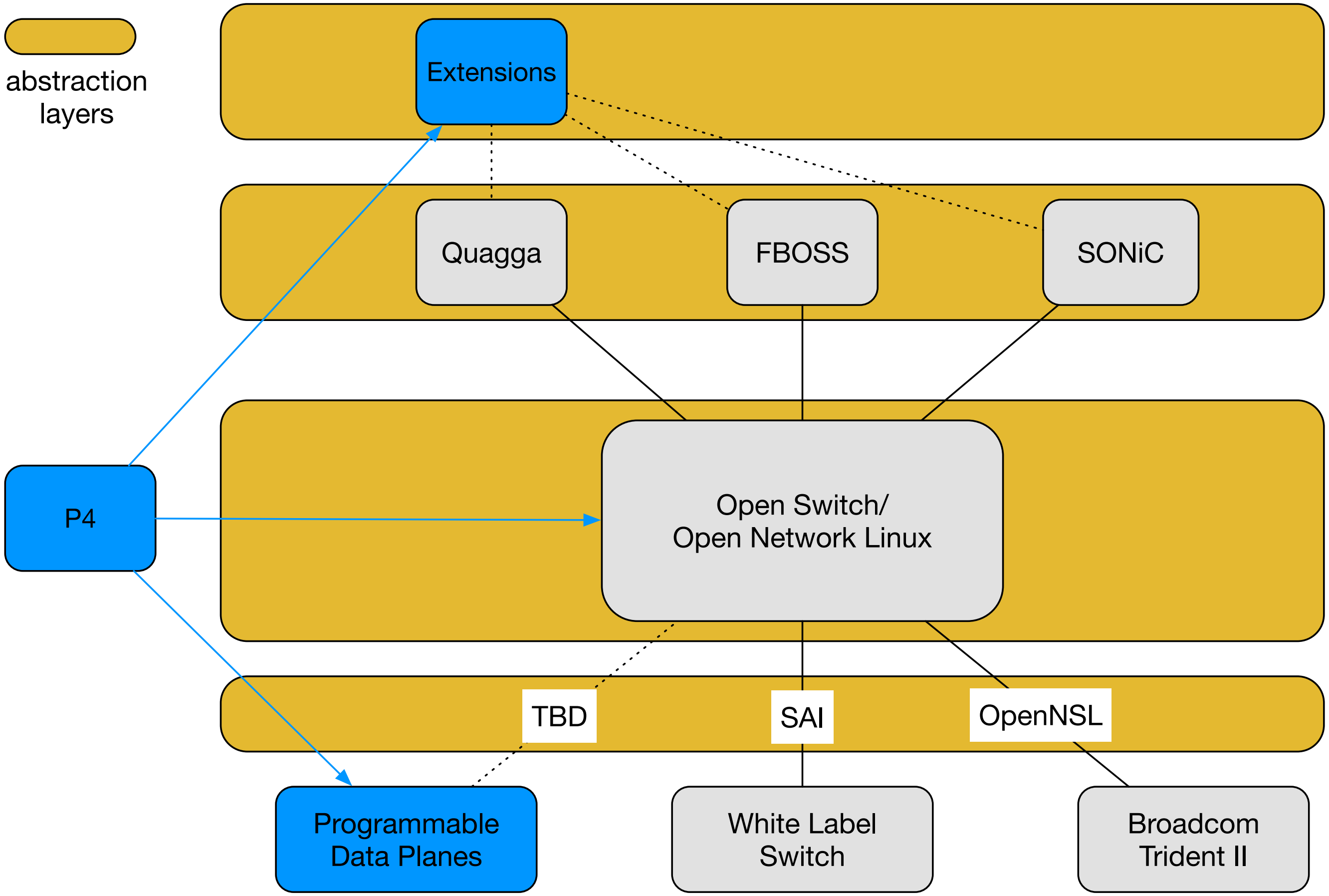
P4 Control Flow

- If/else
- +, *, -, <<, >>, &, |, ^
- ~, -
- OR, AND
- >, >=, ==, <=, <, !=

Work Flow

- Write P4 program, typically these source files:
  - foo.p4
  - headers.p4
  - parser.p4
- Convert P4 program to JSON configuration
- Load JSON configuration on P4 switch

Network Abstraction Layers



Summary

- OpenFlow started the networking disaggregation
- Many companies have joined the networking disaggregation efforts
- Many open hardware vendors
- Several open source network operating systems and related ecosystems
- Various new programmable network silicon is emerging, TO DO:
  - fit this silicon in the open NOS ecosystems
  - work on design of open APIs and *network abstractions*

Thank You

Ronald.vanderPol@surfnet.nl

Ronald.vanderPol@rvdp.org

https://www.rvdp.org

@rvdpdotorg