Switch programmability

7/2018

Switch Programmability

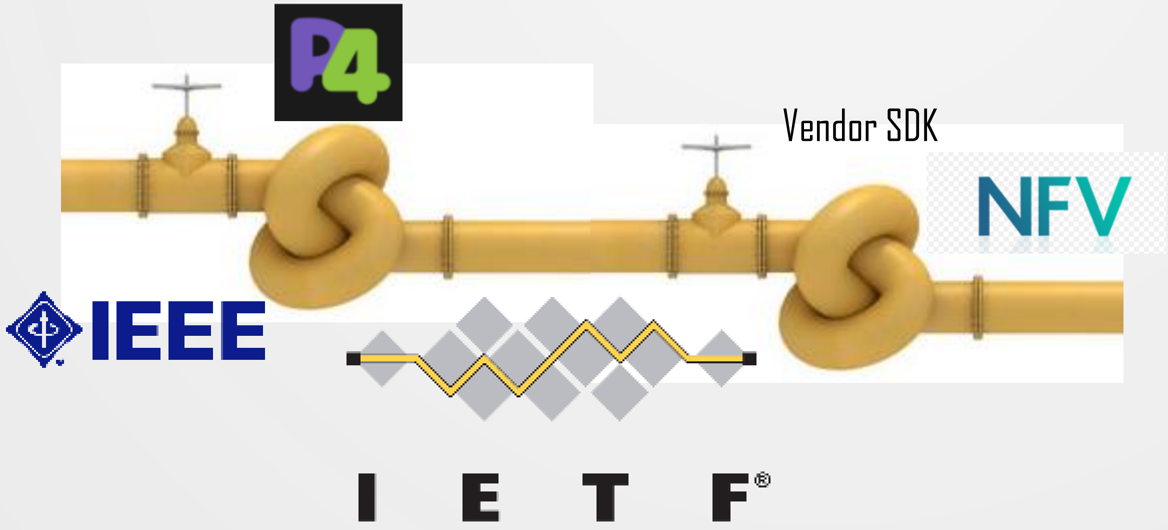
Old world

$$$ on legacy protocols
Best performance and stability
Low feature velocity

Fully Programable

Write everything from scratch
Implement both standard and new applications
Variant feature velocity

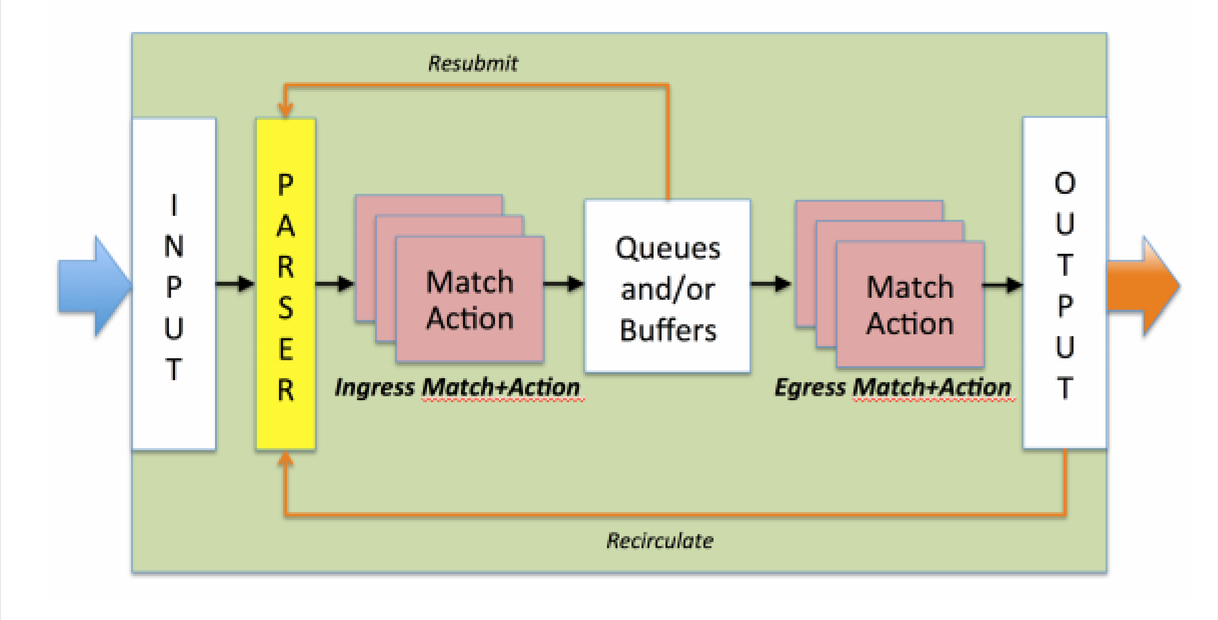
Hybrid



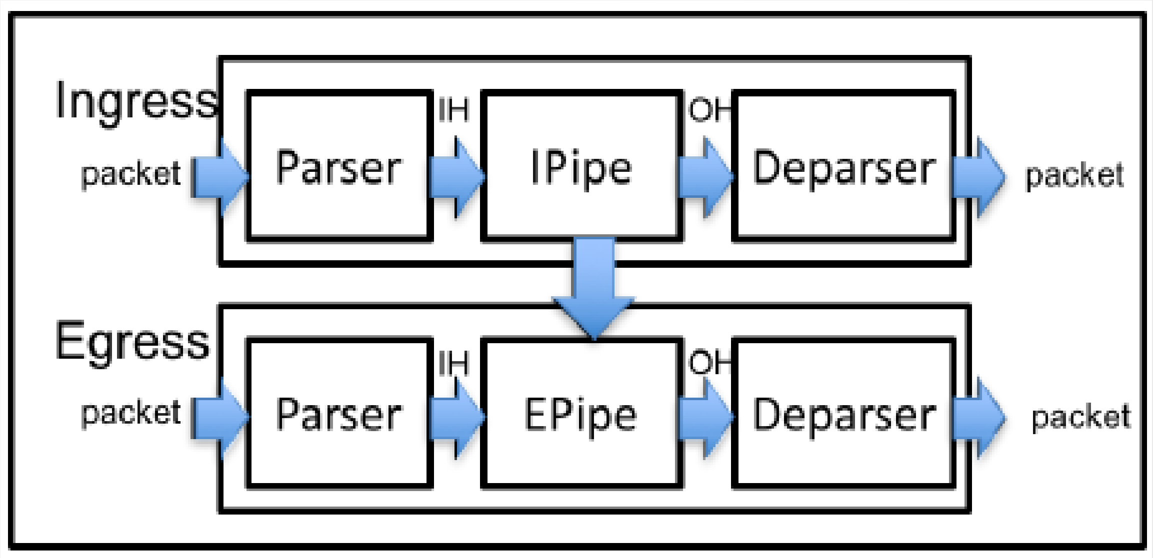
Legacy protocols don't change
Application sand box for home grown needs
Extended HW longevity
High feature velocity

Introduction to P4

V14

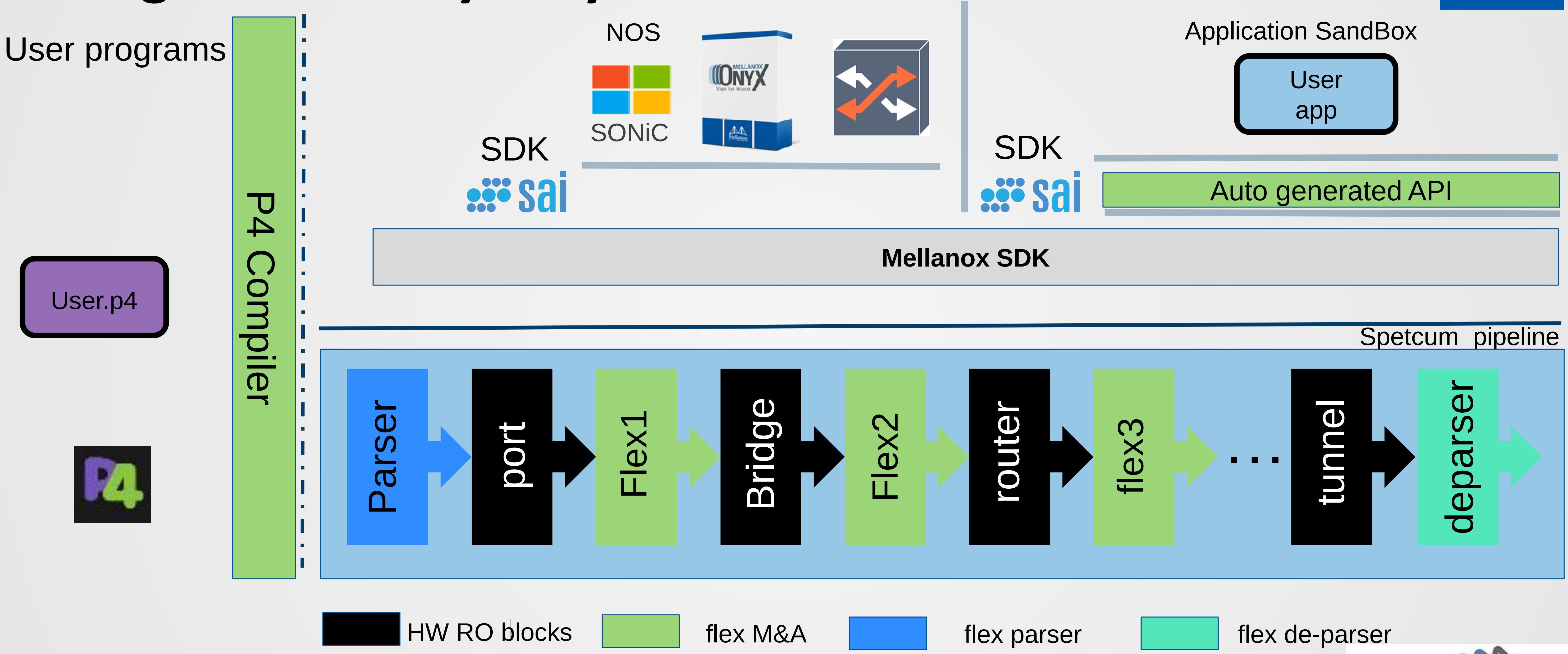


V16



From the spec:
- Introducing P4 architecture description language
- "The P4 architecture can be thought of as a contract between the program and the target"

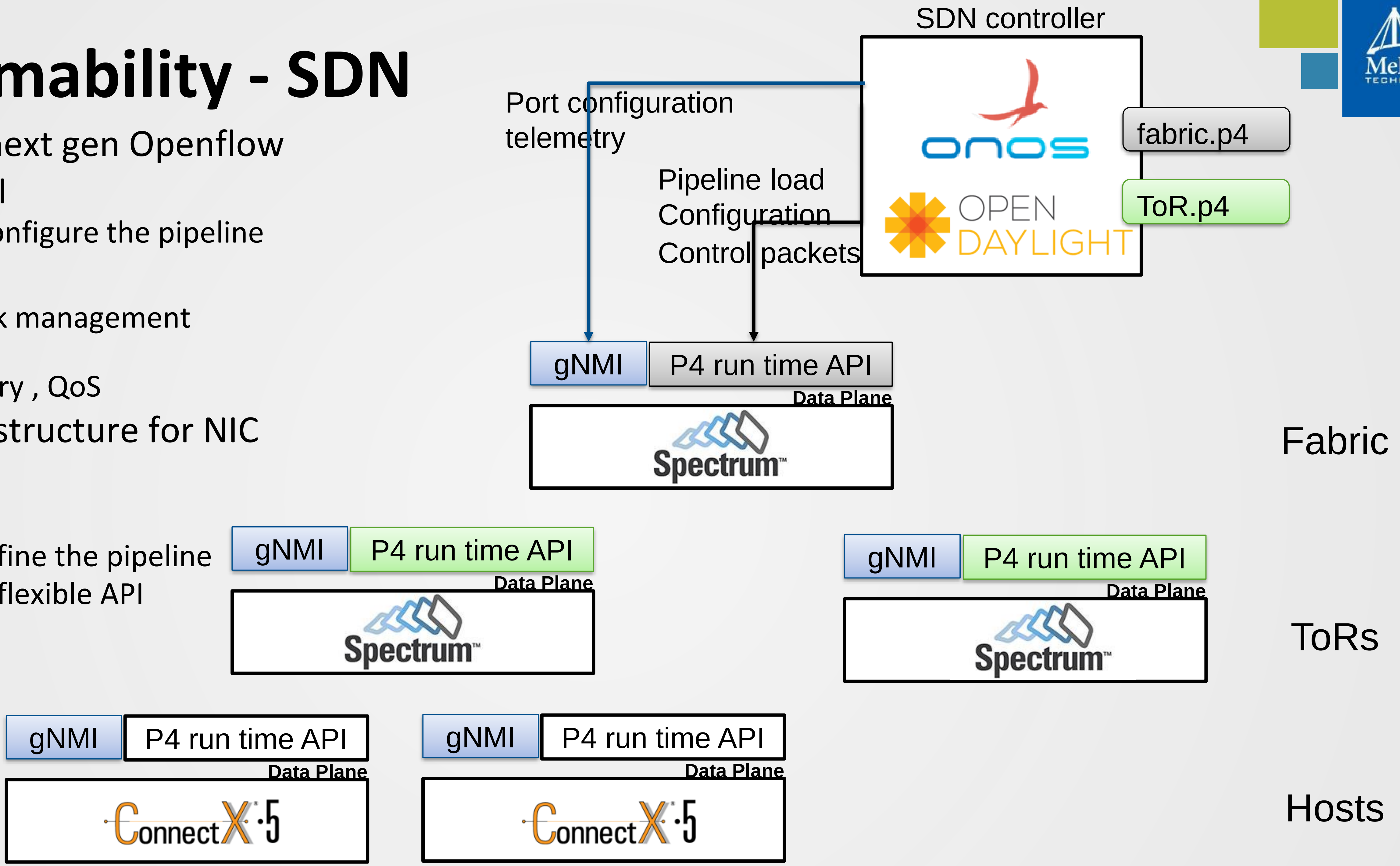
Programmability – Hybrid



Multiple switching SW options, develop apps not NOS

Programmability - SDN

- P4runtime = next gen Openflow
- P4runtime API
  - Define and configure the pipeline
- gNMI
  - gRPC network management interface)
  - Port ,telemetry , QoS
- Uniform infrastructure for NIC and Switch
- True SDN
  - Controller define the pipeline
  - Uniform and flexible API
  - Flexible



Demos

7/2018

BMToR

Adding Bare Metal services to the Cloud

7/2018

Adding Bare Metal services to the Cloud

Goal

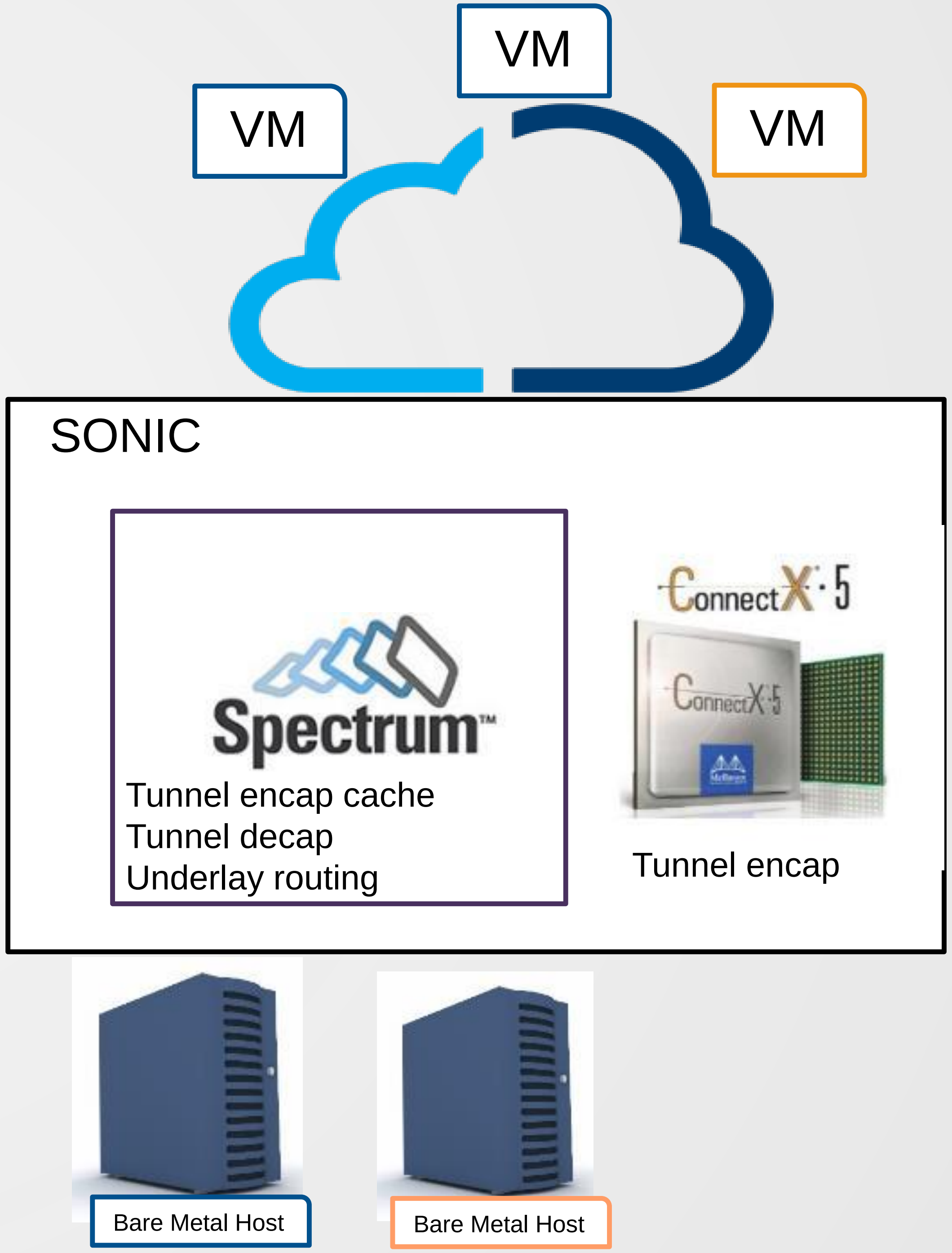
- Connect Bare Metal machine to cloud VMs

Challenges

- Scalability

Solution

- Programmable pipeline implementation for encapsulation logic

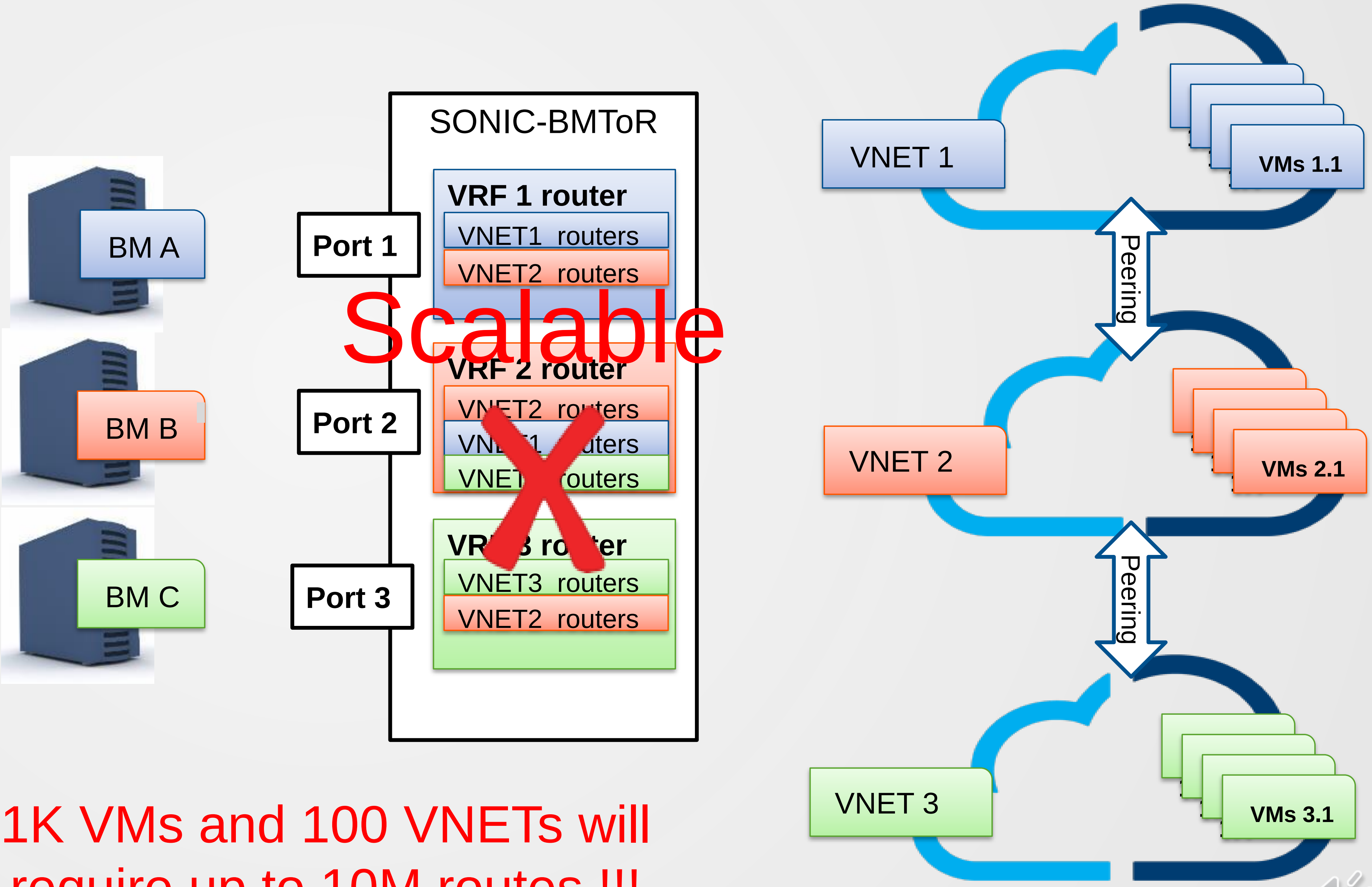


VNET peering in Legacy network

- VNET-virtual network
- VNET peering -Peering between virtual networks

Implementation:

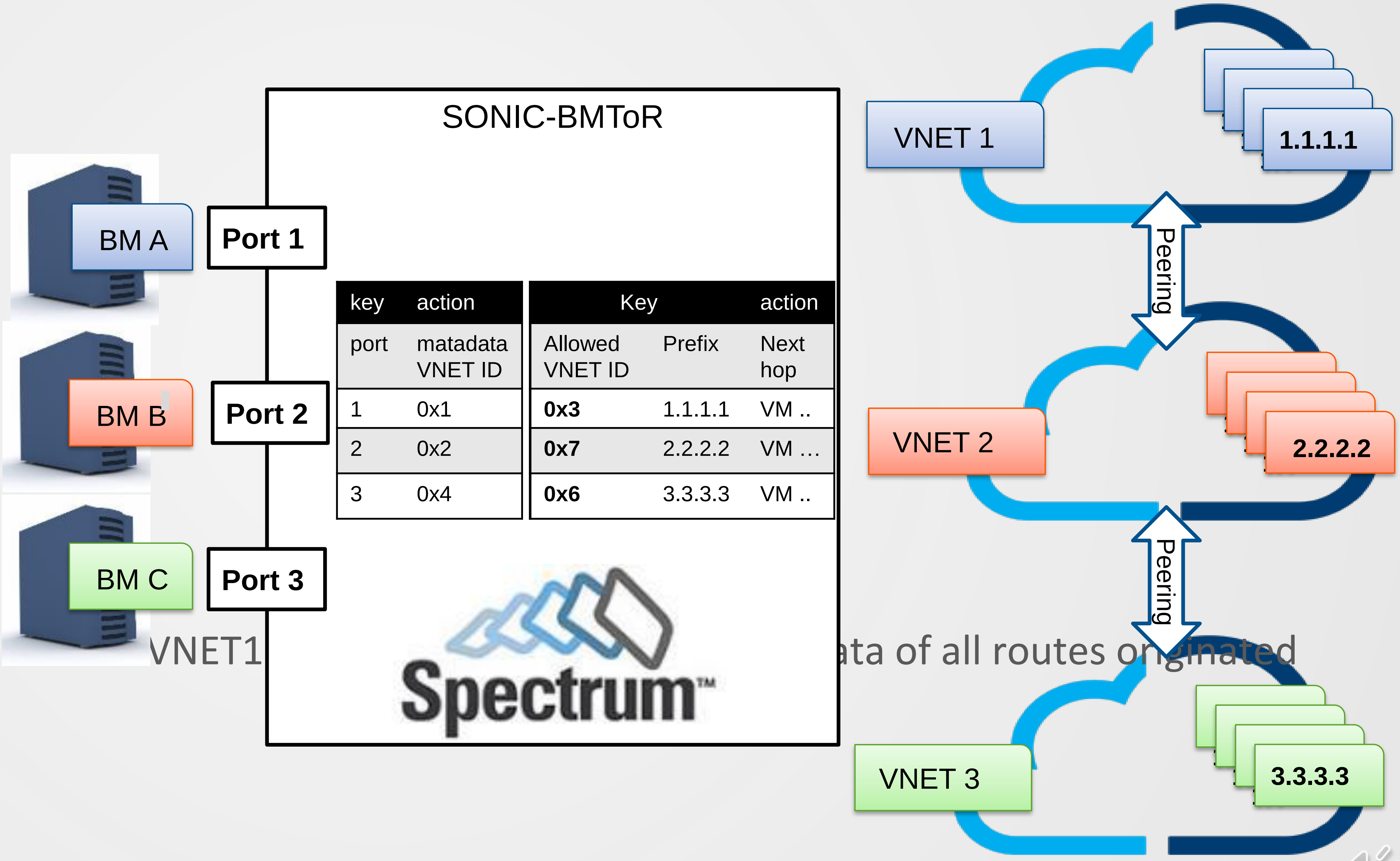
- VNET -> VRF
- VNET1 peering with VNET2 -> copy route from VNET1 to VNET2 and vice versa



1K VMs and 100 VNETs will require up to 10M routes !!!

VNET peering in programmable network

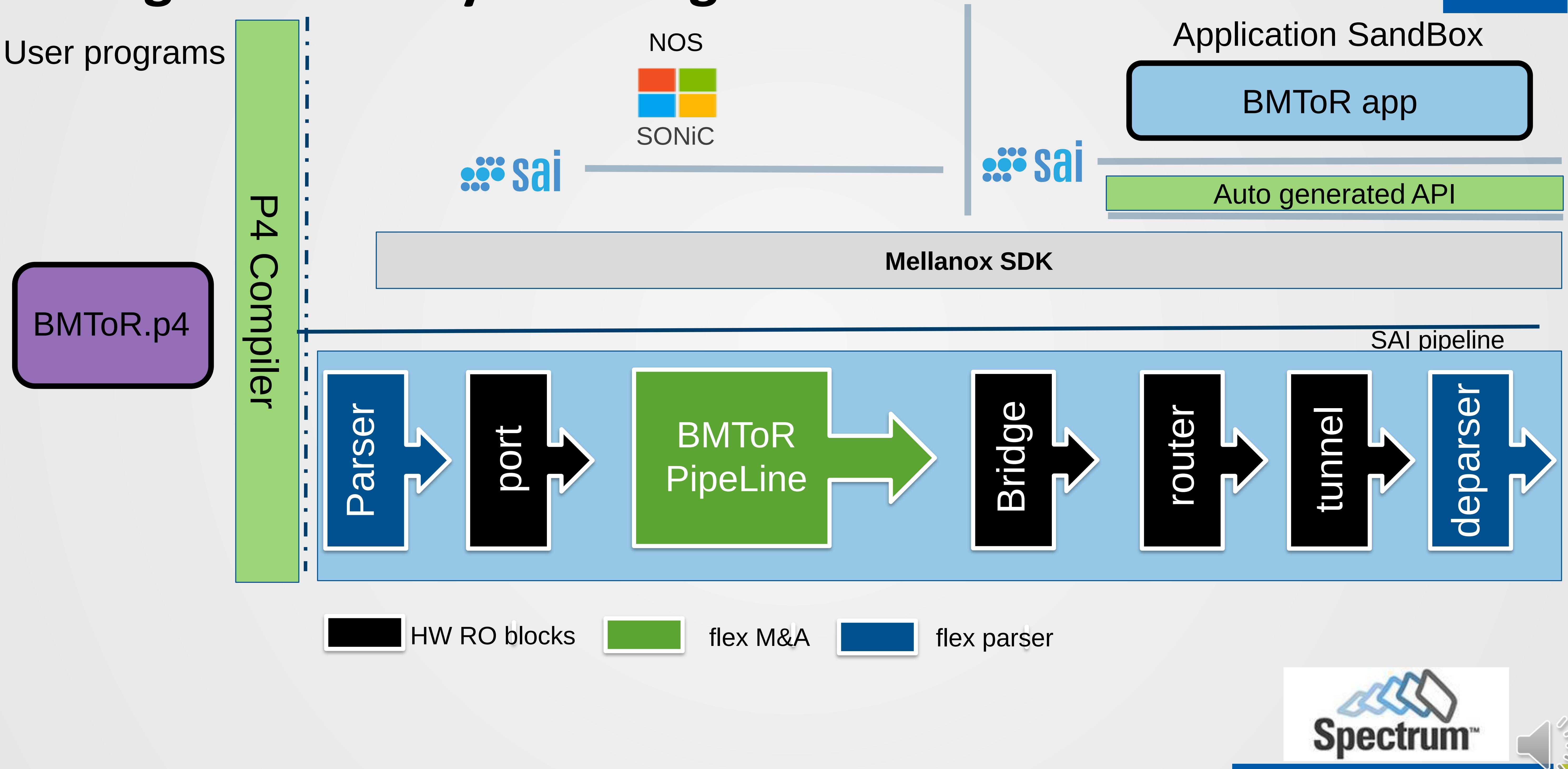
- Two match action tables
- Port to VNET
  - Key: Port
  - Action Set metadata
    - metadata = VNET ID
- VNET routing
  - Key: metadata , prefix
    - metadata vector of VNET peers
  - Action: next hop
- VNET1 peering with VNET2 -> VNET1 ata of all routes originated by VNET2
  - A single route per VM
  - Single update per VM route



key	action
port	matadata VNET ID
1	0x1
2	0x2
3	0x4

Key		action
Allowed VNET ID	Prefix	Next hop
0x3	1.1.1.1	VM ..
0x7	2.2.2.2	VM …
0x6	3.3.3.3	VM ..

Programmability – adding BMToR

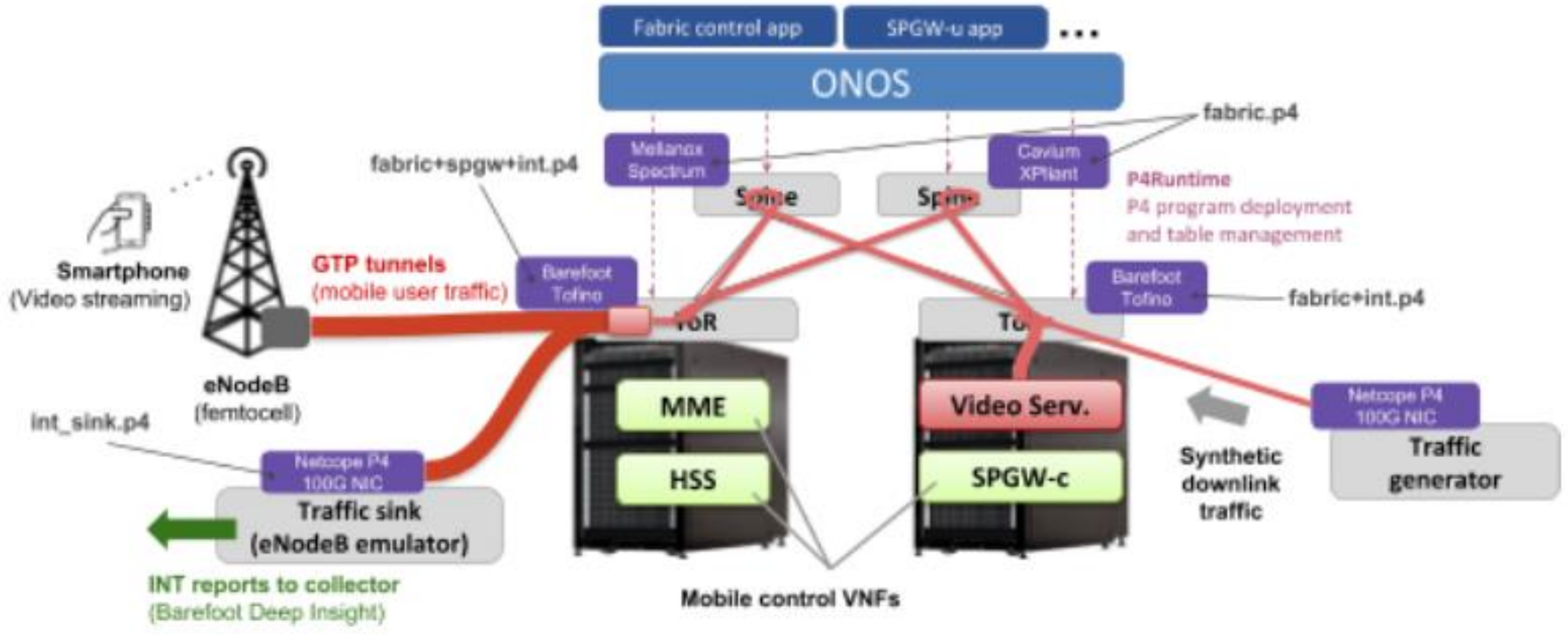


P4Runtime demo with CORD/ONOS

7/2018

ONOS/CORD integration

- Jan 2018 – Integrated at ONL
  - Mellanox Spectrum driver added to ONOS release 1.13
    - https://github.com/opennetworkinglab/onos/releases/tag/1.13.1
  - Spectrum/P4 Runtime wiki instructions added to onosproject.org
    - https://wiki.onosproject.org/display/ONOS/Controlling+P4Runtime-enabled+Spectrum+switch+with+ONOS
  - Spectrum demo at MWC Barcelona and ONS LA
    - Featured Spectrum as a fabric spine, next to Cavium spine and 2 BF leaves



Demo architecture

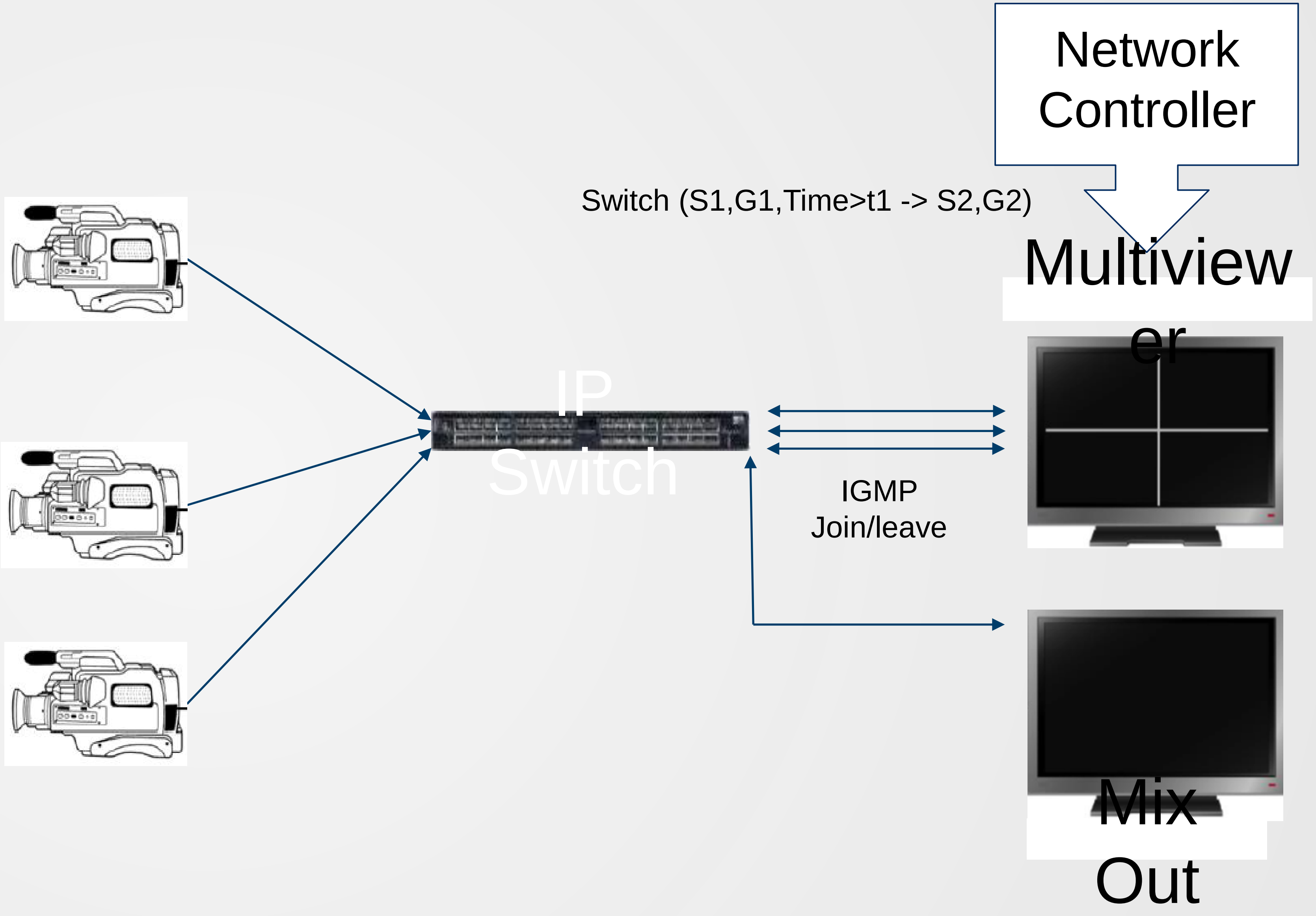
Timed switch over

7/2018

Why is timed switch needed?

Current Solution:

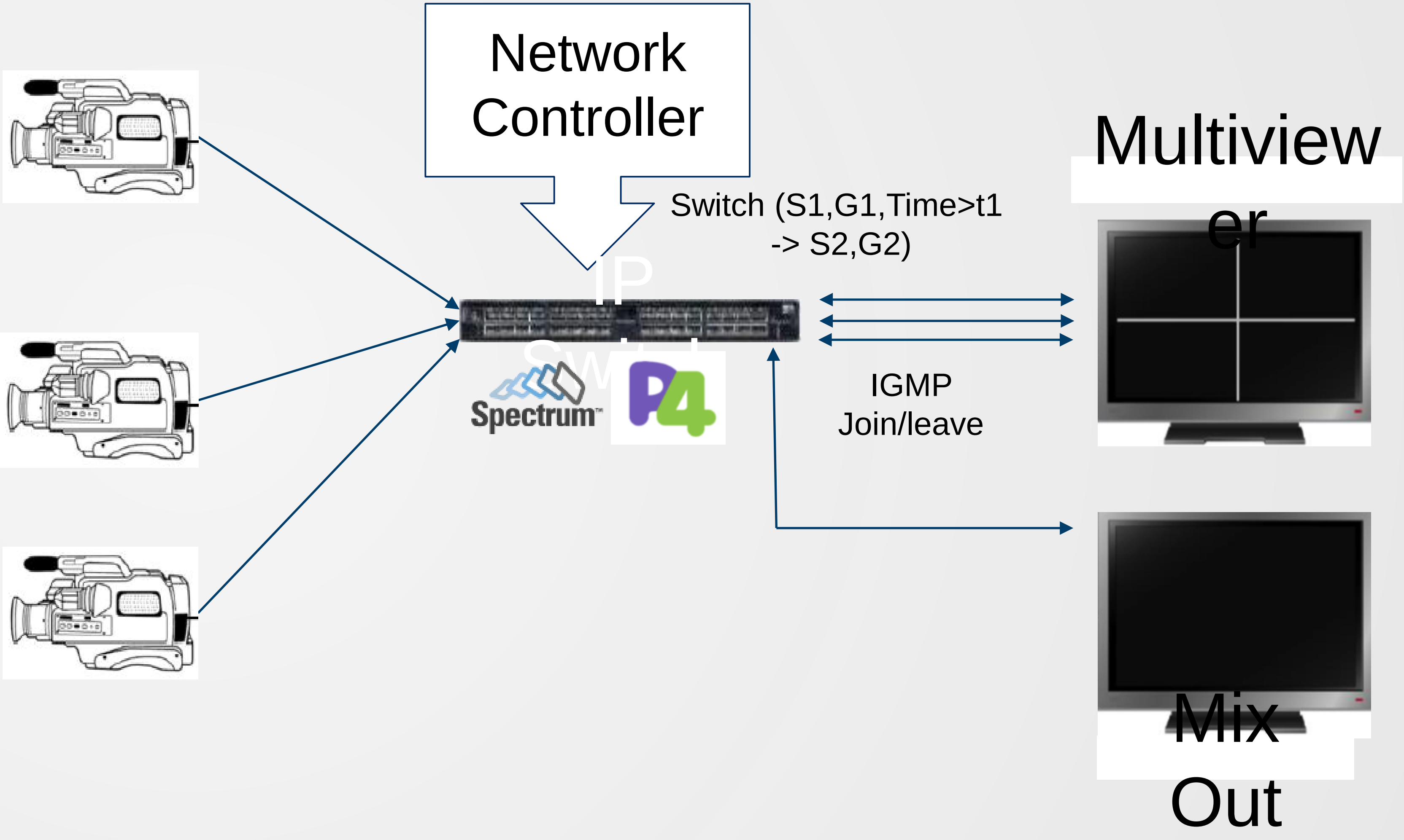
- The endpoint need to make a ‘clean’ switch between different media streams
  - Clean = switch the stream at the frame boundaries
- IGMP based implementation:
  - Use IGMP at the endpoint to join the new flow while receiving the old one
  - Buffer both streams at the endpoint and switch at the frame boundary to the new stream
  - IGMP leave the old flow
- Down side
  - **Endpoint link needs to reserve BW for both old and new streams**
  - Endpoint buffer need to have room for both streams
  - Latency due to buffer size



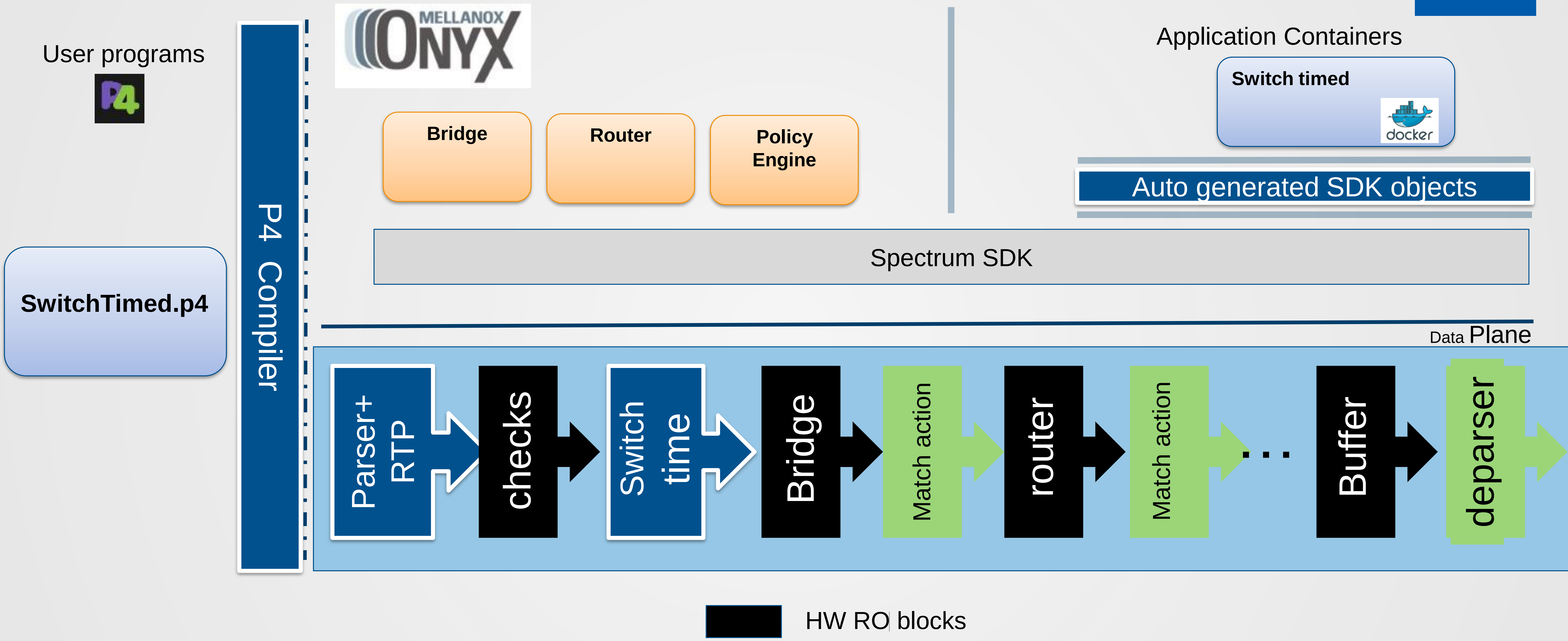
Why is timed switch needed?

Spectrum Programmable Pipeline Solution:

- Timed switch implementation:
  - Match on RTP timestamp on received media streams
  - All media flow time stamps are synchronized/ locked. All packets from the same frame carry the same stamp
  - Switch between flows at the new timestamp value (exact match or regex)
- Advantages
  - Programmable hybrid pipeline: All the legacy protocols (IGMP/ PTP/ PIM/…) are operational along the per flow timed switch implementation
  - Network/endpoint links carries only relevant data i.e. link can be utilized to carry more streams
  - Reduced frame buffer and latency at the endpoints



Spectrum Programmable Hybrid Pipeline



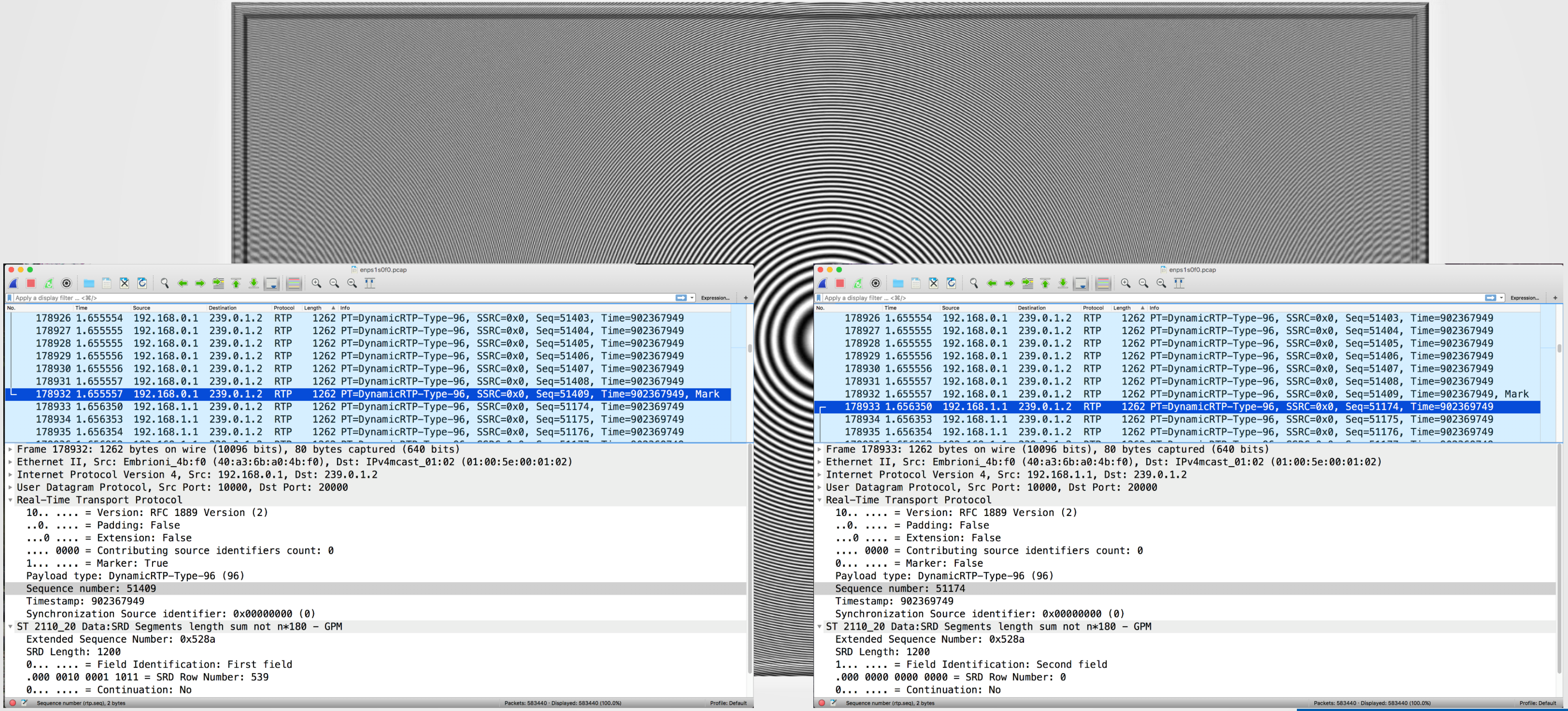
- Hybrid – the integration between legacy (switch router) and programmable pipeline
- NOS (ONYX) and user applications run in parallel

P4 timed switch/ salvo program

```
    table table_timestamp {
        key = {
            headers.rtp.timestamp : range;
        }
        actions = {set_range_bitmap;}
        size = 256;
    }
    table table_ip_mc_forward{
        key = {
            standard_metadata.METADATA_REG : ternary;
            headers.ip.v4.dst_addr : exact;
            headers.ip.v4.src_addr : exact;
        }
        actions = {to_ports;}
        size = 256;
    }
    // pipe
    apply{
        table_timestamp.apply();
        table_udp_port.apply();
        table_ip_mc_forward.apply();
    }
}
control control_in_rif(inout Headers_t headers, inout metadata_t meta, inout standard_metadata_t standard_metadata){
    apply{}
}
control control_out_rif(inout Headers_t headers, inout metadata_t meta, inout standard_metadata_t standard_metadata){
    apply{}
}
control control_out_port(inout Headers_t headers, inout metadata_t meta, inout standard_metadata_t standard_metadata){
    apply{}
}
SpectrumSwitch(
    SalvoParser(),
    control_in_port(),
    control_in_rif(),
    control_out_rif(),
    control_out_port(),
    SalvoDeparser()
    ) main;
```

Timed Switch Demo

Switch between 2 streams on frame boundary, every 5 seconds



Thank You