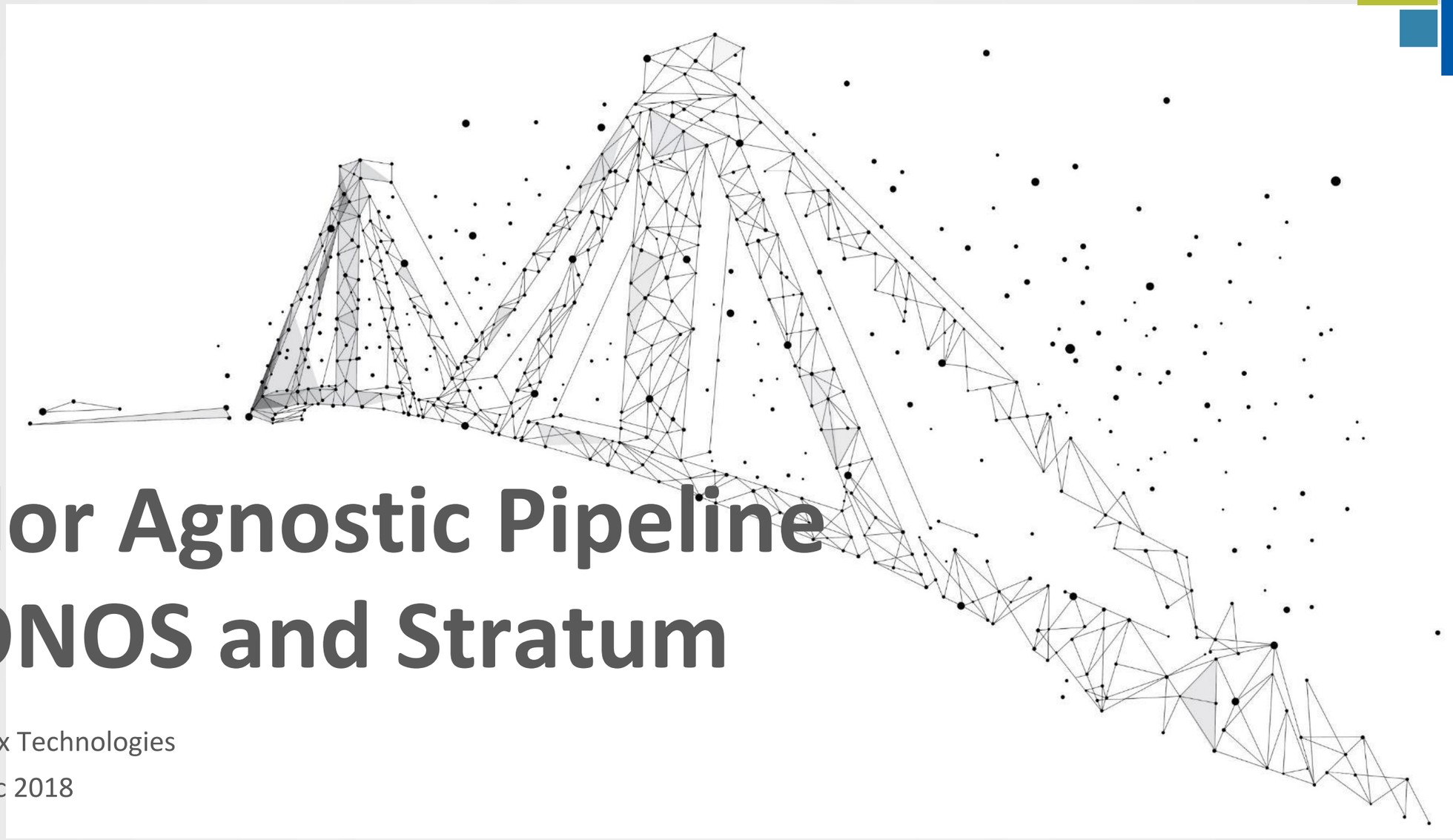


Vendor Agnostic Pipeline For ONOS and Stratum

Alan Lo, Mellanox Technologies

ONF Connect Dec 2018

A Generic Modular Switch Architecture

About Us @ Mellanox Technologies

- Alan Lo, Senior SW architect
  - Advanced Development Group, Switch Architecture
  - P4 Compiler backend for Spectrum/Spectrum 2
  - P4 Runtime architecture and prototyping
- Matty Kadosh, Principle Architect
  - All things involving “Switch Programmability”

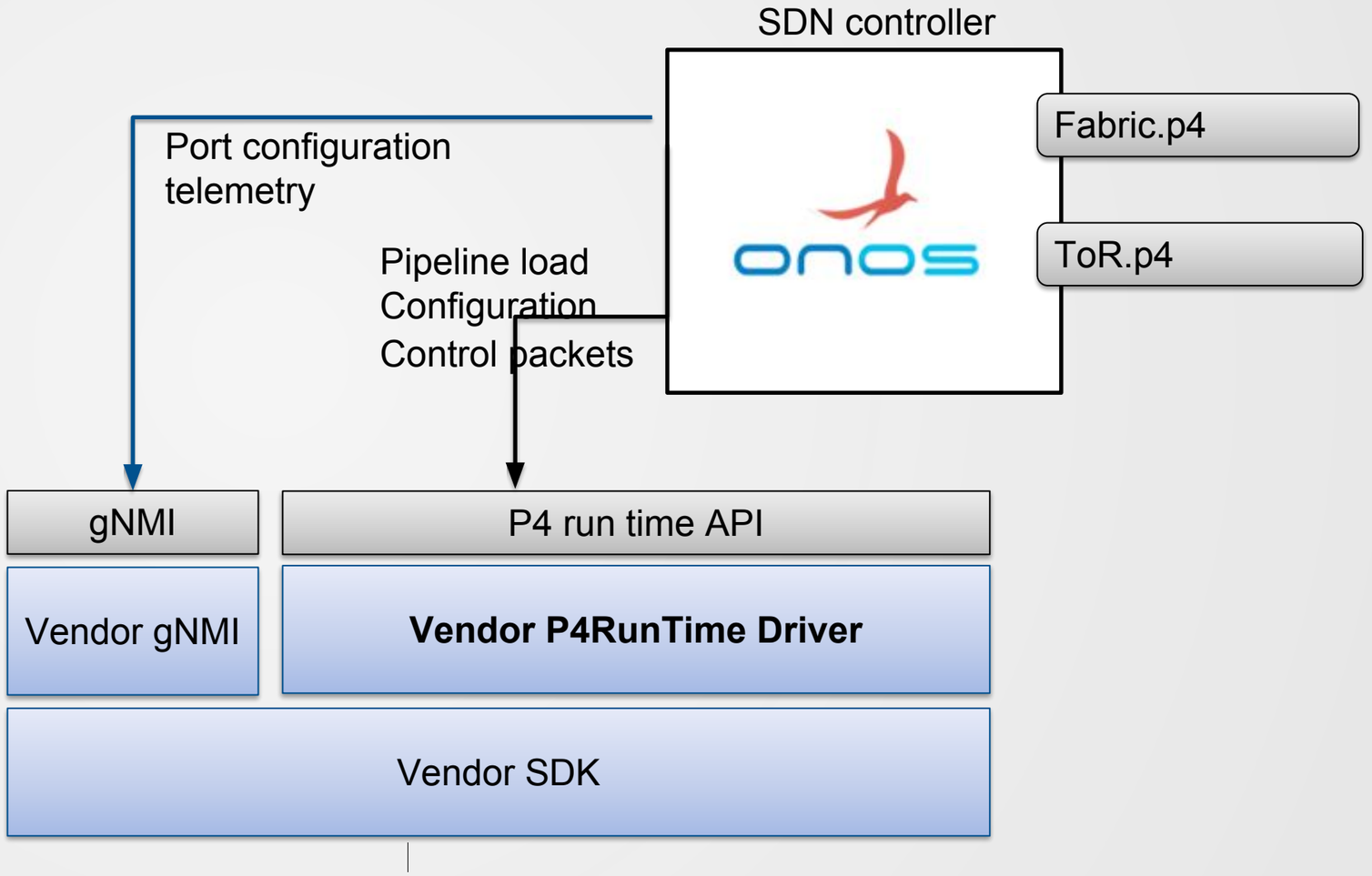
.

Our Motivation for a P4 GMSA target architecture

- Most of the current P4 programs are based on the v1Model -> requires substantial effort to map to other ASICs
- Legacy functionality such as routing, bridging, ILM, rewritten in P4
- A common pipeline (such as the one described by SAI.p4), with legacy functionality already defined by the community, includes many switching hardware vendors!
- A common P4 Runtime agent can be implemented using SAI as the southbound API that will be ASIC vendor agnostic and NOS agnostic (Sonic, Open Switch, and potentially Stratum).
- Programmable functionality can be introduced using a recent contribution to SAI called flex-SAI.
  - framework enables the P4 programmer to enhance the existing SAI pipeline
  - e.g. gateway functionality, in-band telemetry, custom parser and match action etc.

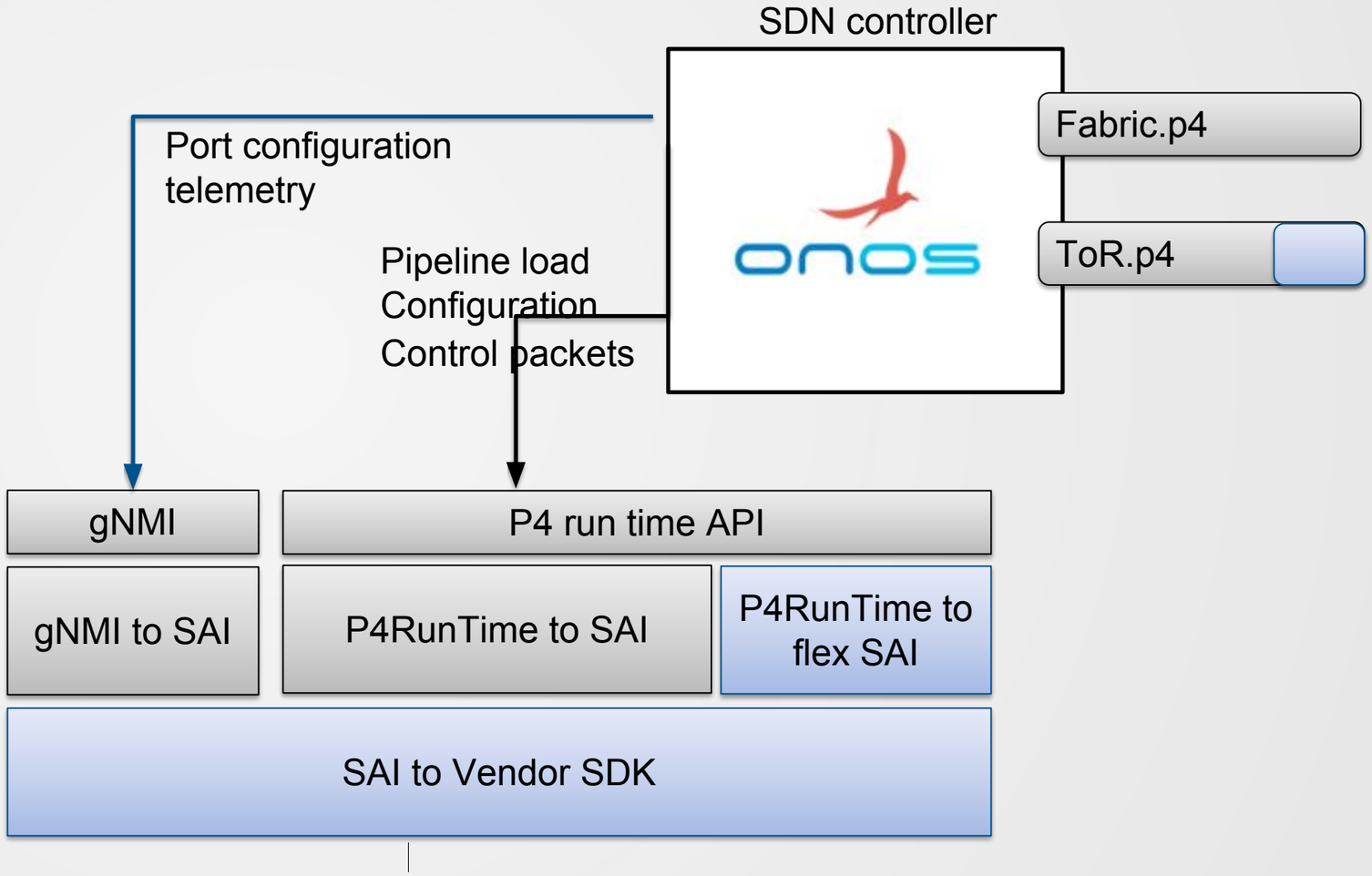
ONOS –P4RunTime current view

- P4 program are base on V1model
- Monolithic Switch
- Need “reinvent the wheel “ and write bridge and router …
- Designed to be BMV2 Software pipeline

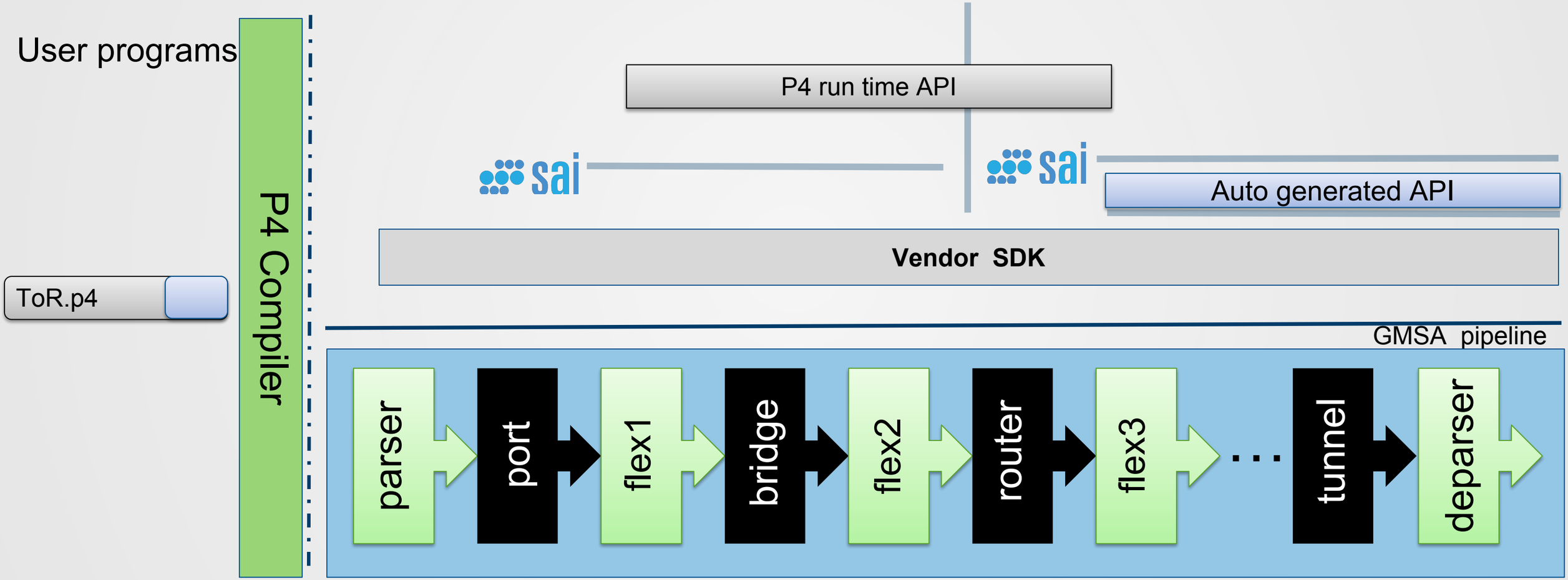


P4 target architecture

- GMSA Motivation
  - vendor agnostic :- e.g. based on SAI behavioral model supported by most switching hardware vendors (Broadcom, Cisco, Mellanox, Marvel, Barefoot, Innovium, Centec, Nephos)
  - NOS agnostic (Sonic, Open Switch, and potentially Stratum).
  - Flexible & Extendible
  - Hybrid - provides an ability to run legacy L3/L2 … and SDN controller in an hybrid mode

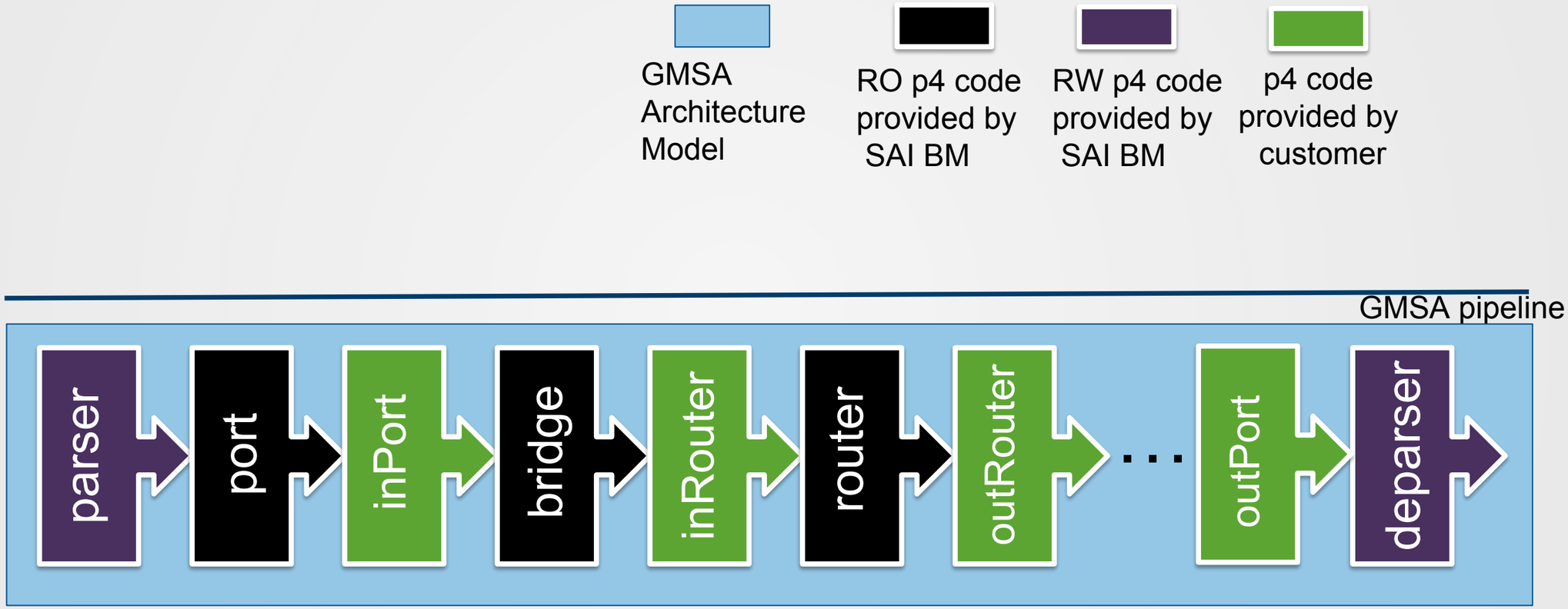


NOS Hybrid example



Generic behavioral model

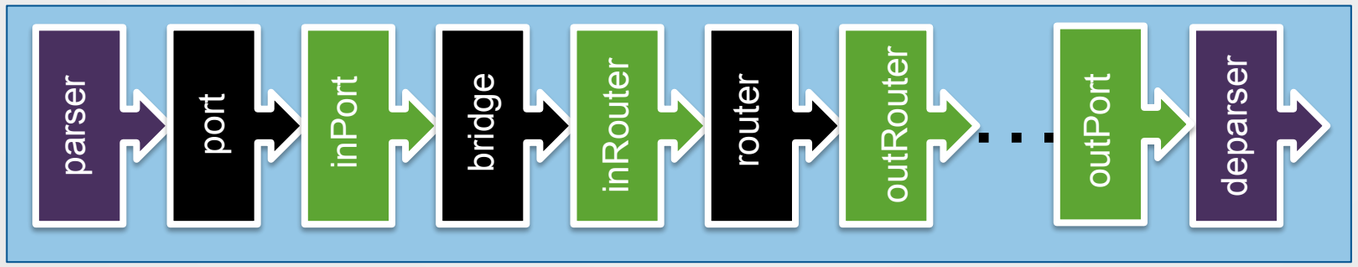
GMSA target Architecture



GMSA target Architecture

```
package GMSA(
    parser ,
    in_port,
    in_router,
    out_router,
    out_port,
    deparser );
```

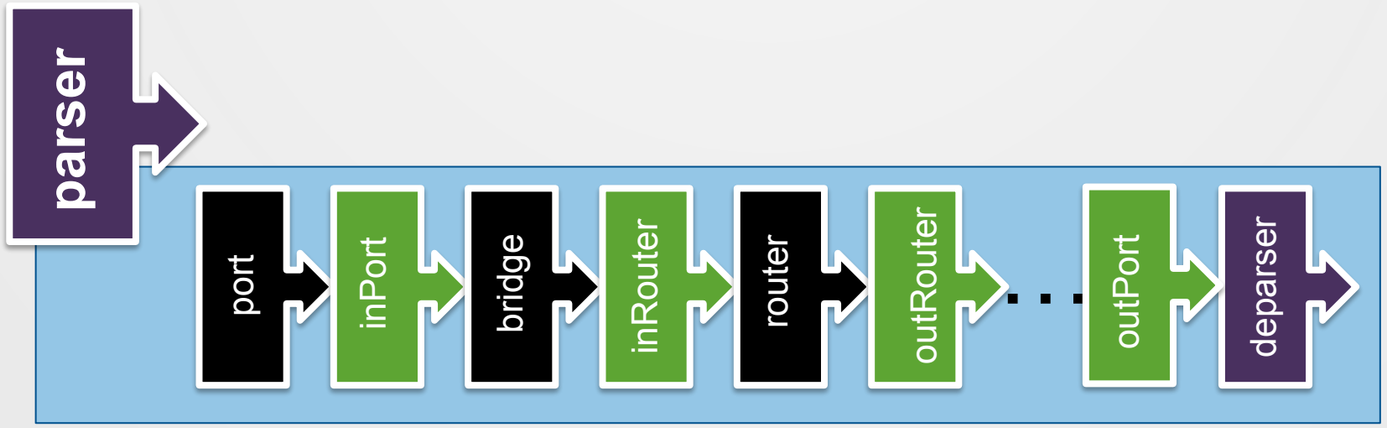
- GMSA_header.p4 – SAI header definition
- GMSA_parser.p4 - Provides basic parser, can be edited by the user
- GMSA_deparser.p4 - Provides basic deparser, can be edited by the user
- GMSA_action.p4 - SAI supported action, can be extended by vendor
- GMSA_metadata<stage>.p4 - SAI standard metadata action, can be extended by vendor
  - <stage> - generic, inPort, inRouter …



GMSA target Architecture

```
package GMSA(
    parser ,
    in_port,
    in_router,
    out_router,
    out_port,
    deparser );
```

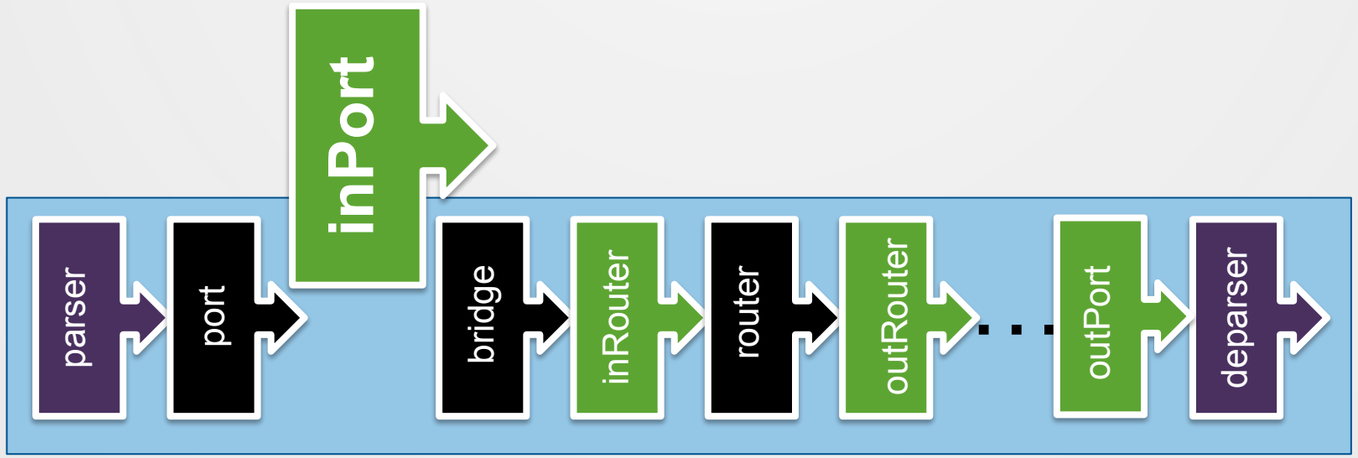
```
control parser(
      packet_in packet ,
          inout generic_meta  g_meta,
          out GMSA_header headers);
```



GMSA target Architecture

```
package GMSA(
    parser ,
    in_port,
    in_router,
    out_router,
    out_port,
    deparser );
```

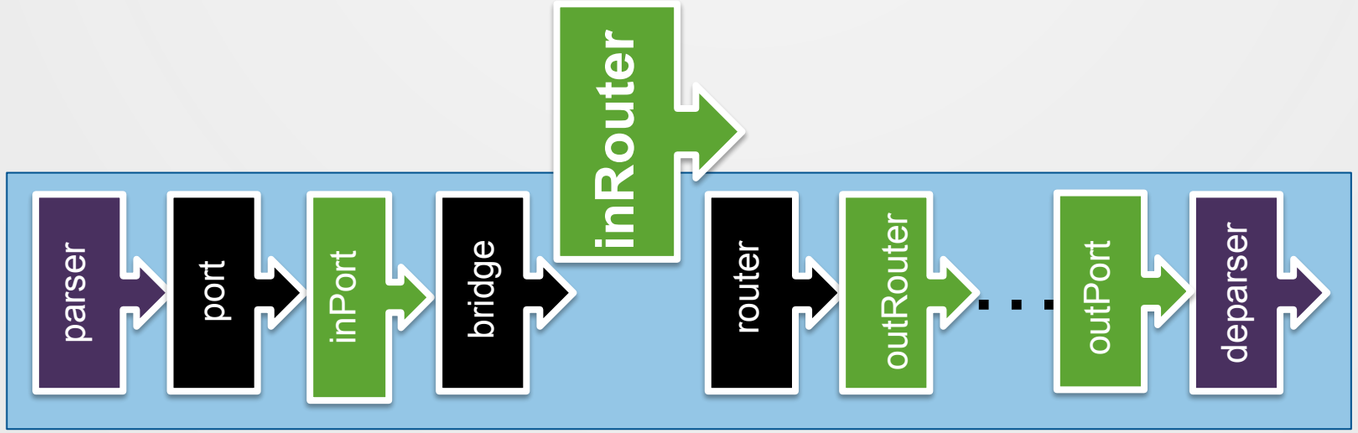
```
control in_port(
    inout GMSA_header headers,
        inout generic_meta  g_meta,
        inout in_port_meta  iport_meta);
```



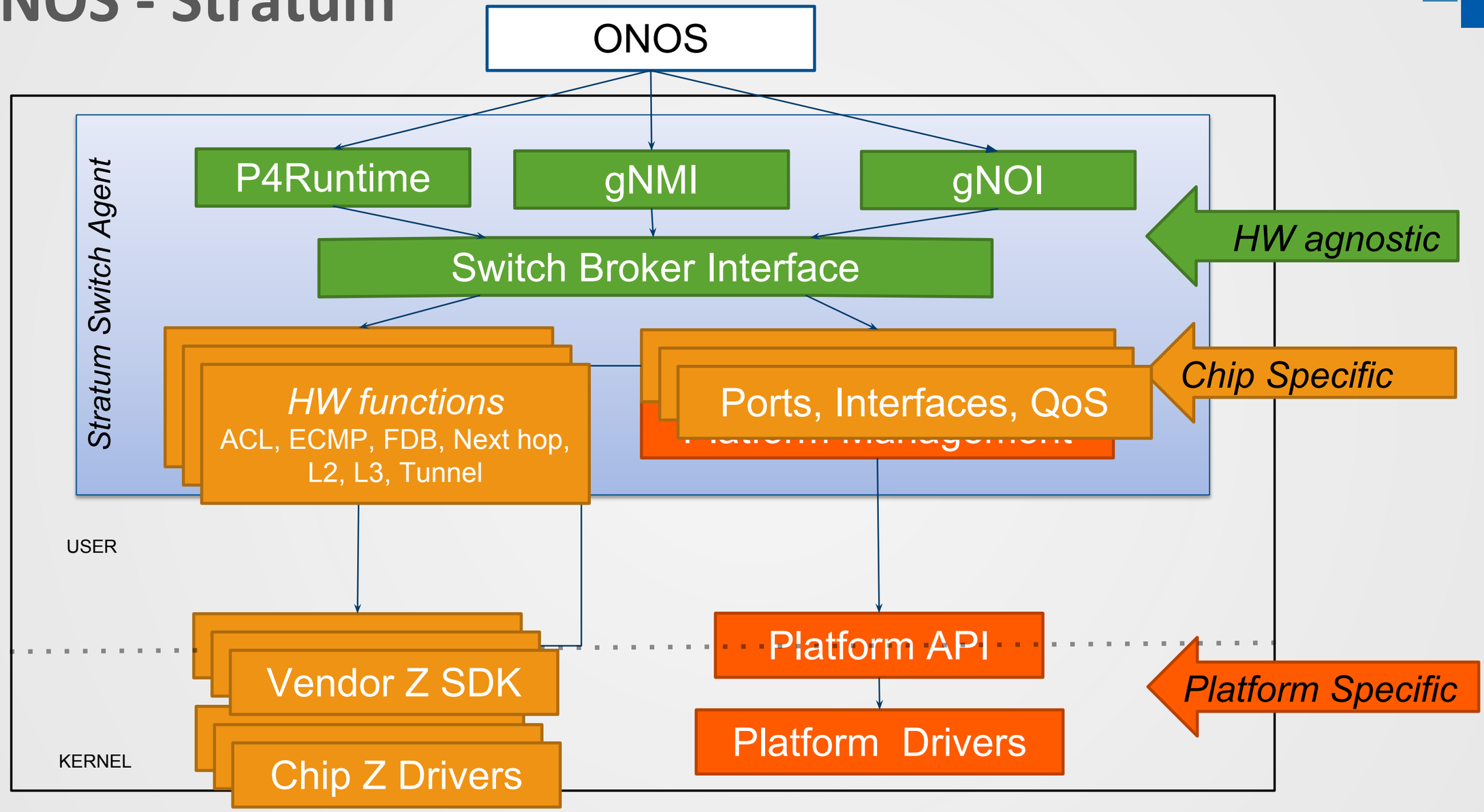
GMSA target Architecture

```
package GMSA(
    parser ,
    in_port,
    in_router,
    out_router,
    out_port,
    deparser );
```

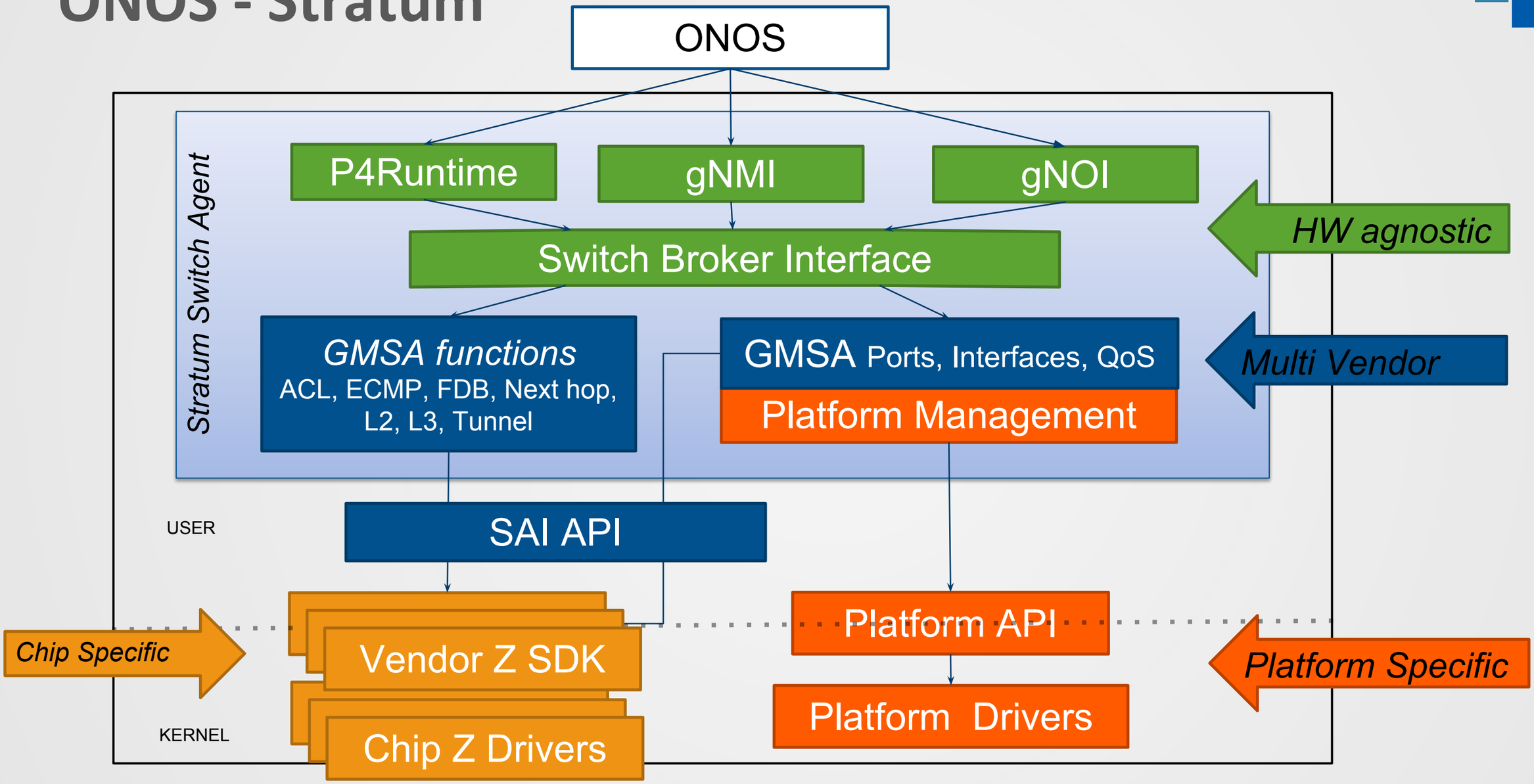
```
control in_router(
    inout GMSA_header headers,
        inout generic_meta  g_meta,
        inout in_router_meta irif_meta);
```



ONOS - Stratum



ONOS - Stratum



Use Case 1:

Adding Bare Metal services to the Cloud

OCP Summit 2018

Adding Bare Metal services to the Cloud

Goal

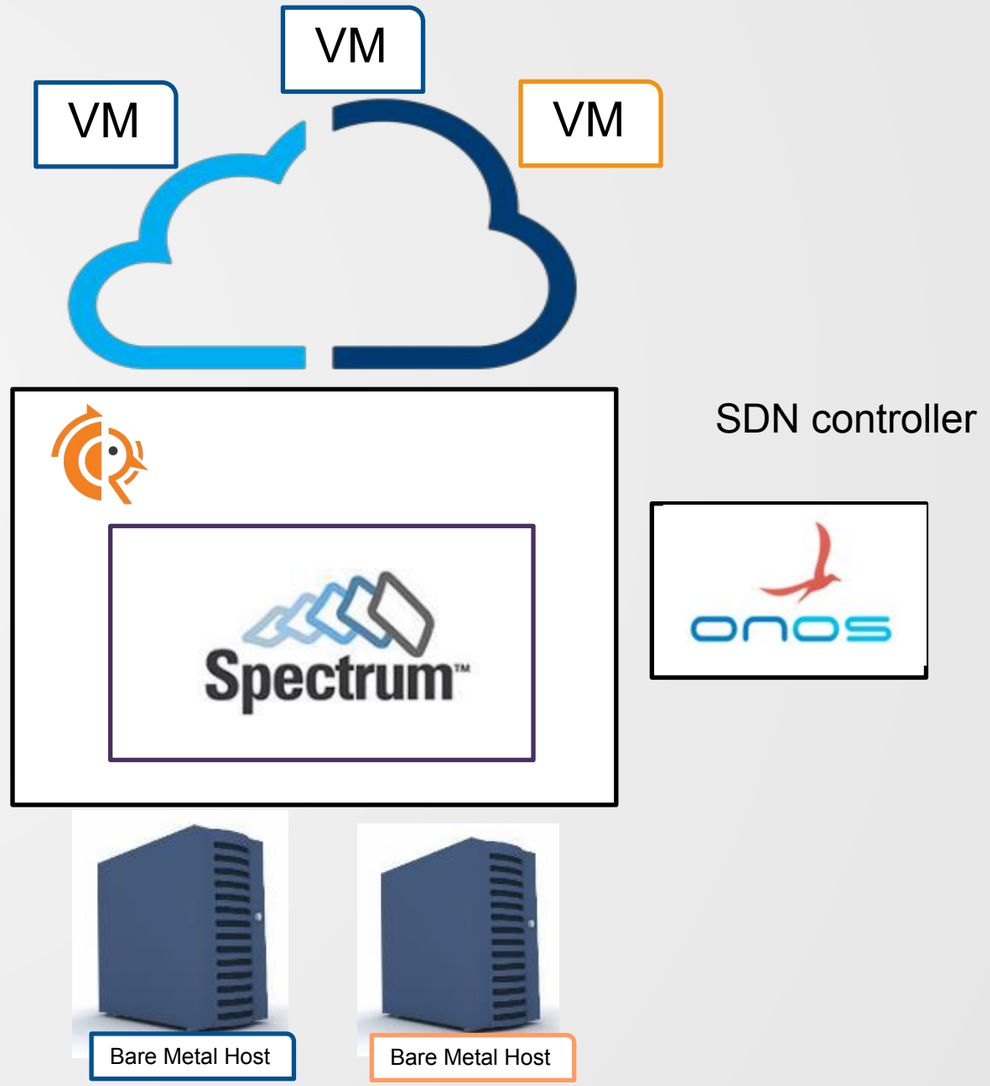
- Connect Bare Metal machine to cloud VMs

Challenges

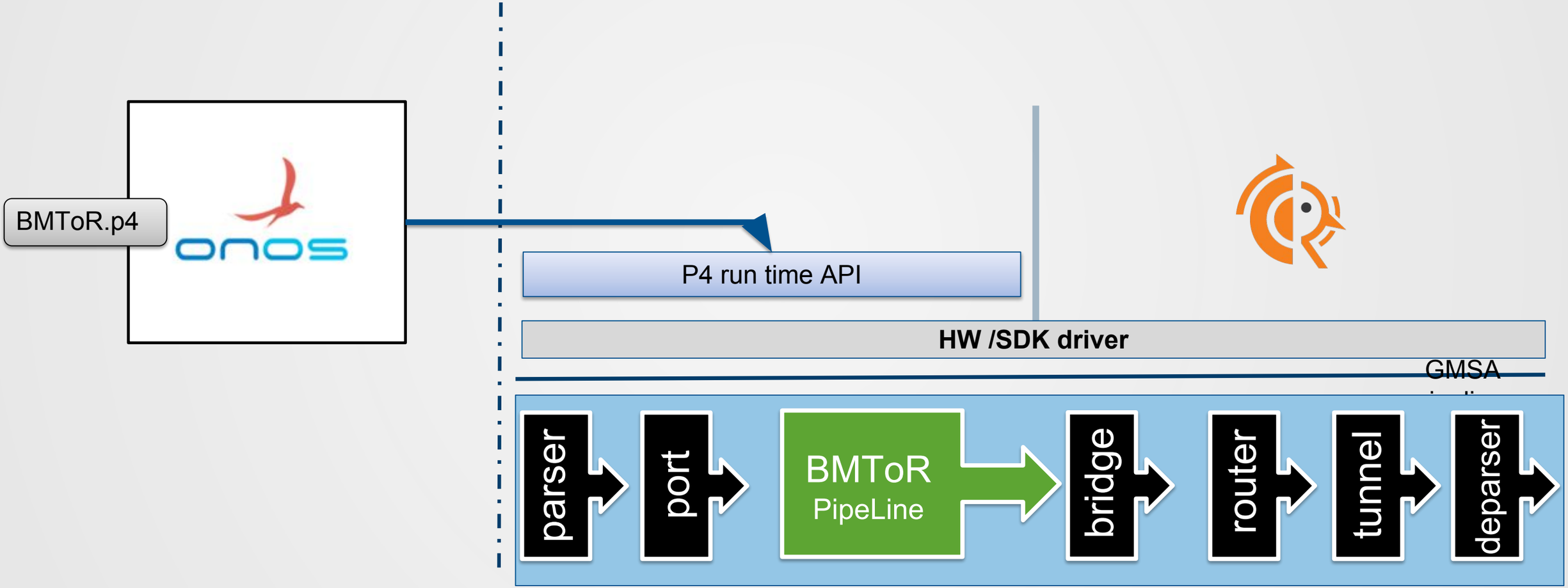
- Scalability
- Hybrid
  - SDN overlay
  - “BGP” underlay

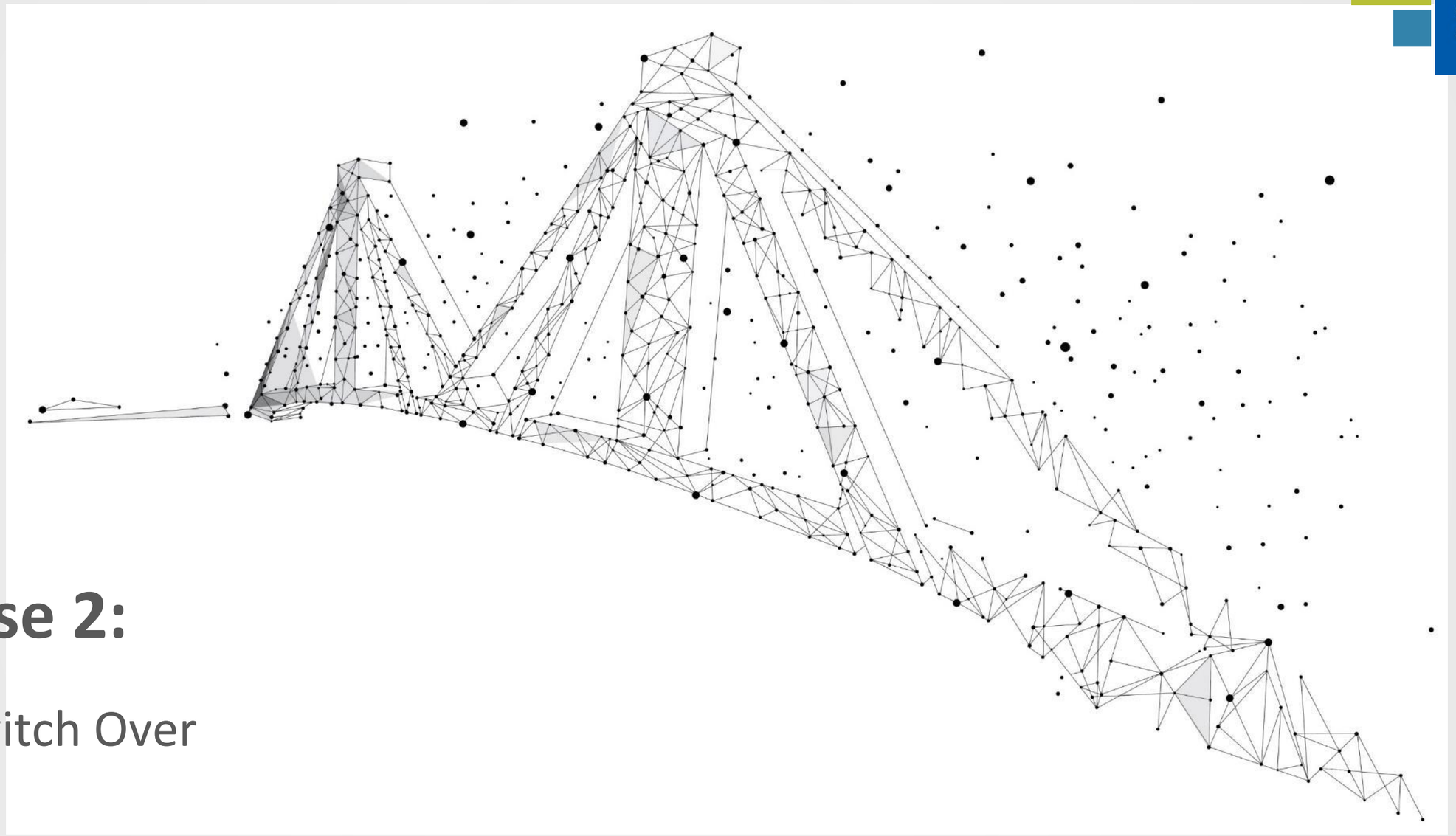
Solution

- Programmable pipeline implementation for encapsulation logic
- ONOS + on box FRR



Adding Bare Metal services to the Cloud





Use Case 2:

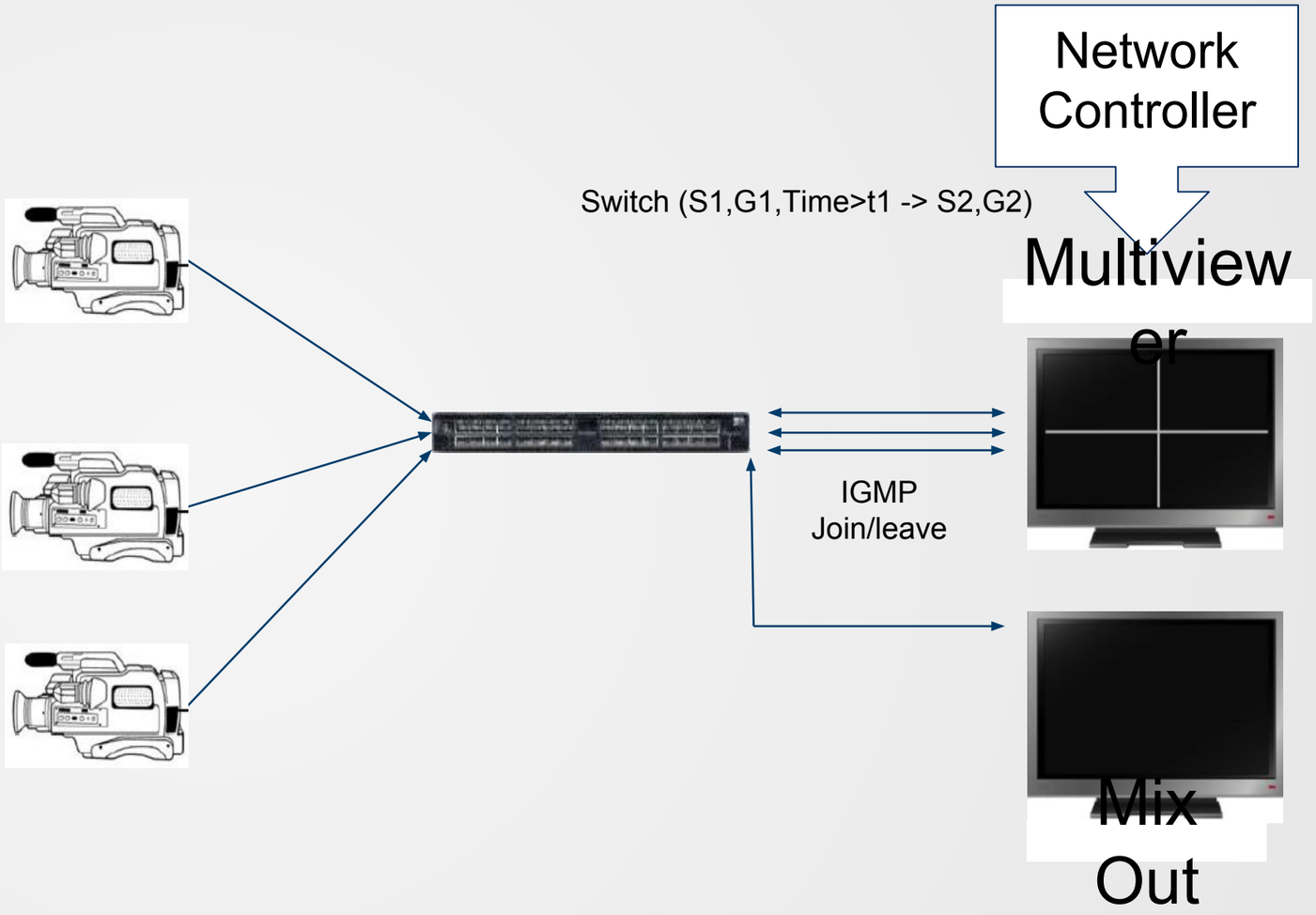
Timed Switch Over

IBC 2018

Why is timed switch needed?

Current Solution:

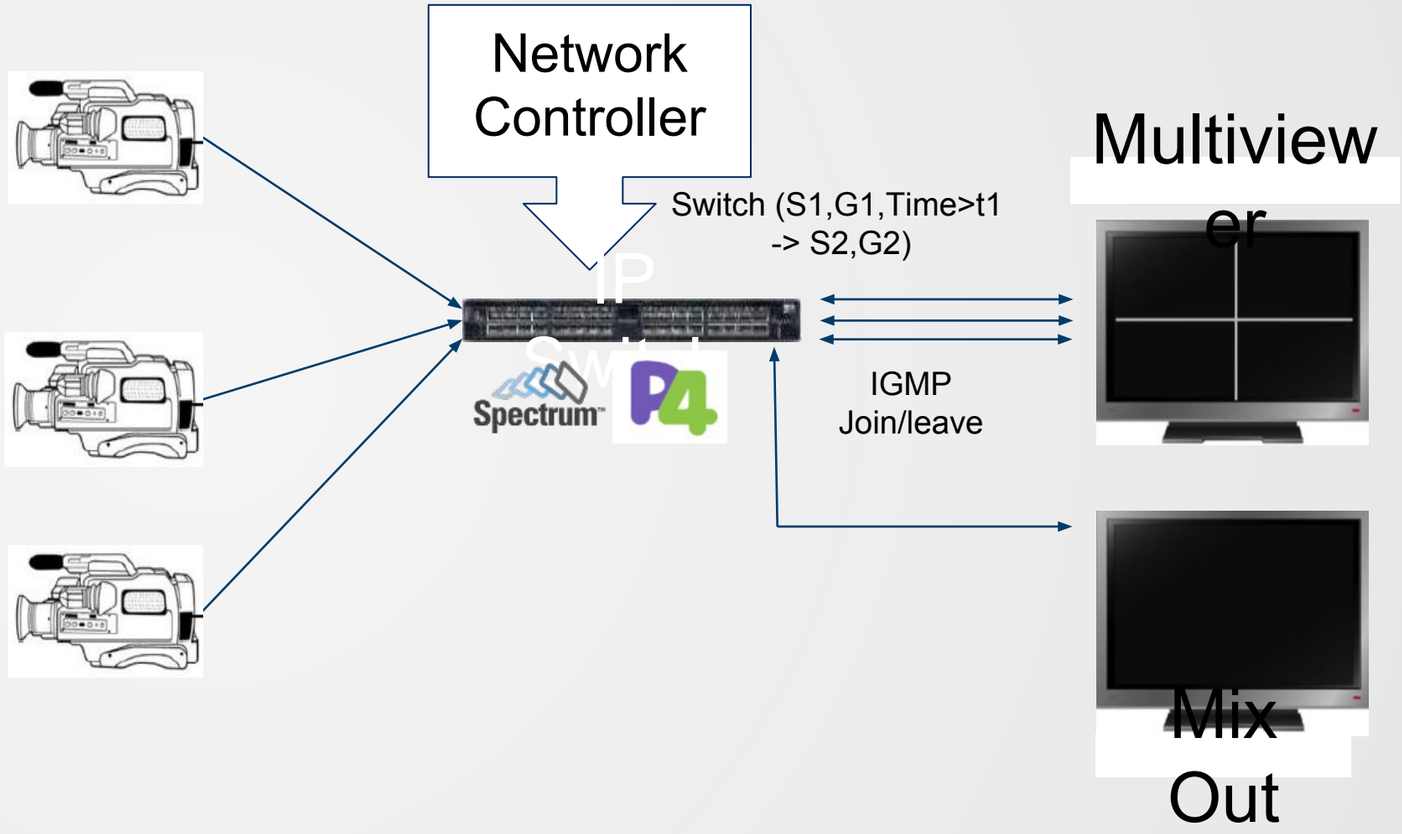
- The endpoint need to make a ‘clean’ switch between different media streams
  - Clean = switch the stream at the video frame boundaries
- IGMP based implementation:
  - Use IGMP at the endpoint to join the new flow while receiving the old one
  - Buffer both streams at the endpoint and switch at the frame boundary to the new stream
  - IGMP leave the old flow
- Down side
  - **Endpoint link needs to reserve 2x BW for both old and new streams**
  - Endpoint buffer need to have room for both streams
  - Latency due to buffer size



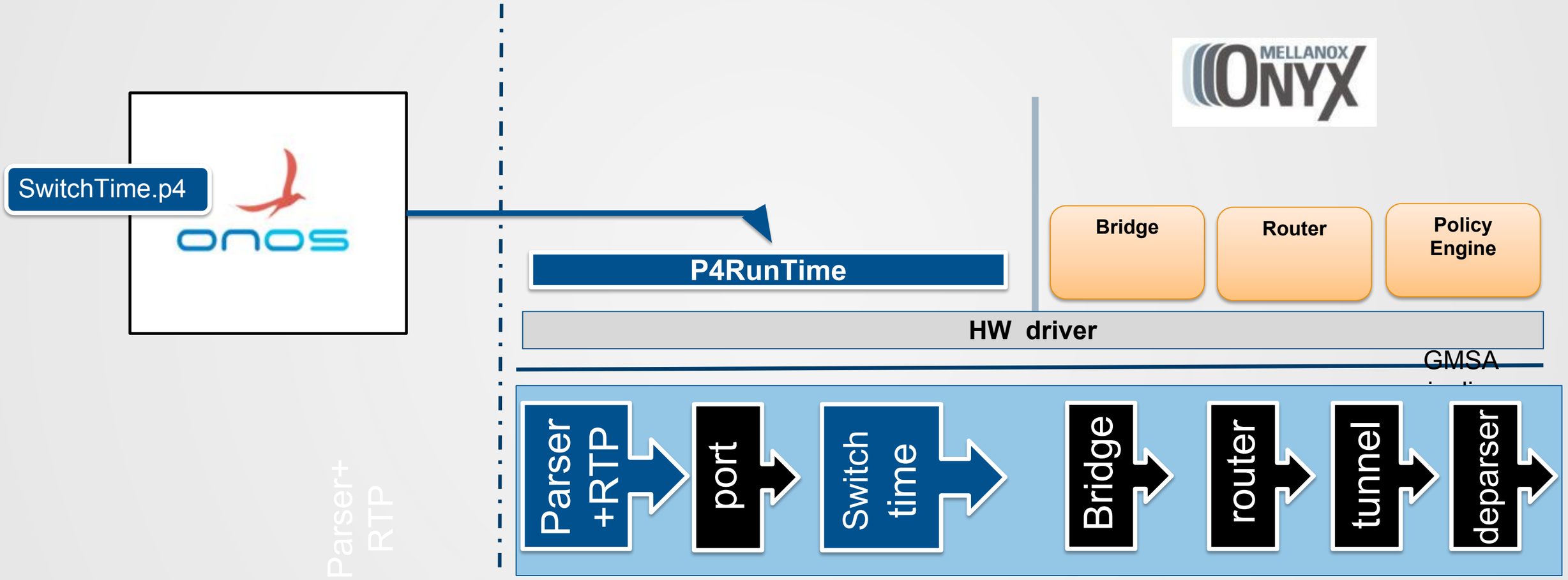
Why is timed switch needed?

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 range match)
- 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



P4 timed switch salvo program

```
control control_in_port(inout Headers_t headers, inout metadata_t meta, inout standard_metadata_t standard_metadata){

    action set_range_bitmap(bit<12> range_bitmap){
        set_meta_reg(range_bitmap, 0x7ff);
    }
    action set_udp_hit() {
        set_meta_reg(0x800, 0x800);
    }
    action set_udp_miss() {
        set_meta_reg(0, 0xfff);
    }
    action to_ports(bit<32> port_pbs_id) {
        set_pbs_port(port_pbs_id);
    }

    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.ipv4.dst_addr : exact;
            headers.ip.ipv4.src_addr : exact;
        }
        actions = {to_ports;}
        size = 256;
    }

    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;
```

P4 BMTor

```
control control_in_port(inout Headers_t headers, inout metadata_t meta, inout standard_metadata_t standard_metadata){

    action set_vnet_bitmap(bit<12> vnet_bitmap){
        set_meta_reg(vnet_bitmap,0x0fff);
    }
    action to_tunnel(bit<32> tunnel_id, bit<32> underlay_dip, bit<16> bridge_id){
        set_bridge(bridge_id);
        vxlan_tunnel_encap(tunnel_id,underlay_dip);
    }
    action to_router(bit<32> router_pbs_id) { // Go to egress router (BM->BM)
        set_pbs_router(router_pbs_id);
    }
    action to_port(bit<32> port_pbs_id) { // Go to DPDK port (Cache miss)
        set_pbs_port(port_pbs_id);
    }

    table table_peering{
        key = {
            standard_metadata.ingress_port :exact;
        }
        actions = {set_vnet_bitmap;}
        size = PORTNUM;
    }
    table table_vhost{
        key = {
            standard_metadata.METADATA_REG : ternary;
            headers.ip.ipv4.dst_addr :exact;
        }
        actions = {to_tunnel;to_router;to_port;}
        size=MSEE_TABLE_SIZE;
    }

    apply{
        table_peering.apply();
        table_vhost.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(
    fixParser(),
    control_in_port(),
    control_in_rif(),
    control_out_rif(),
    control_out_port(),
    fixDeparser()
    ) main;
```