Flexible Ethernet

Old world

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

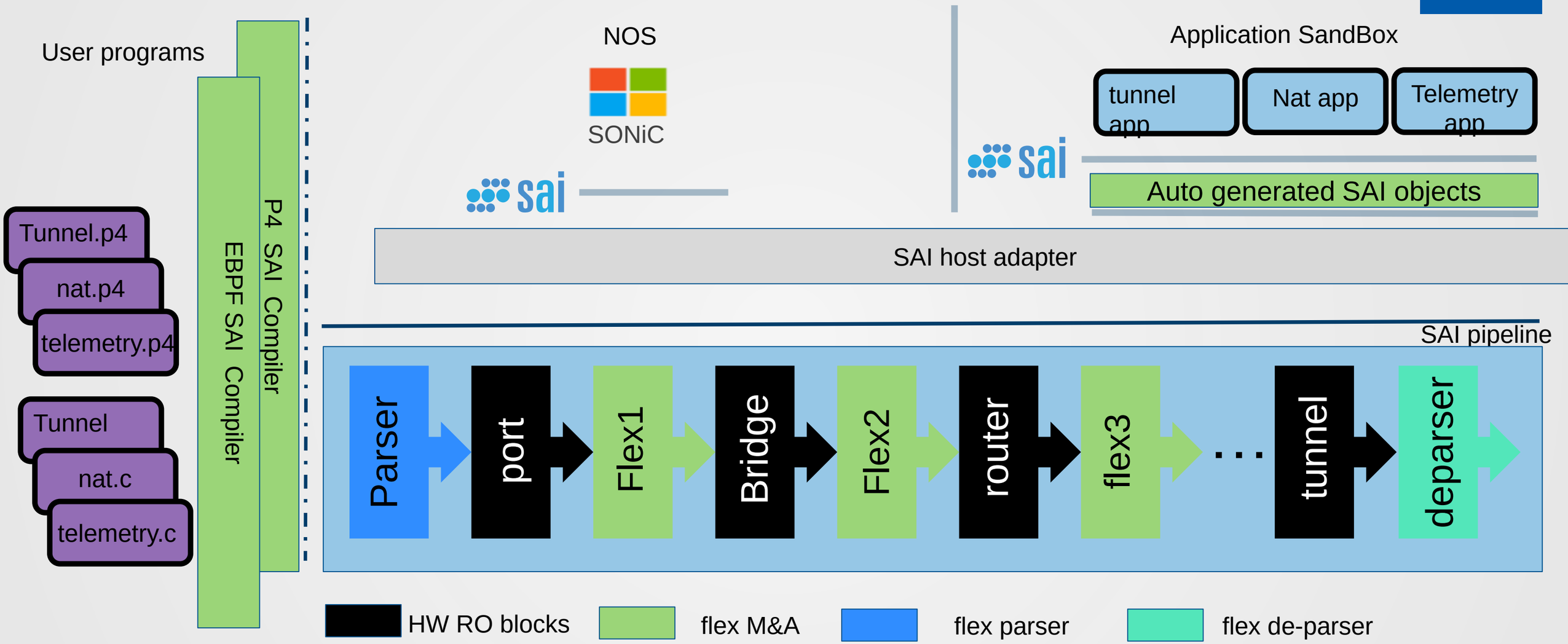
New world?

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

Real world

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

SAI programmability

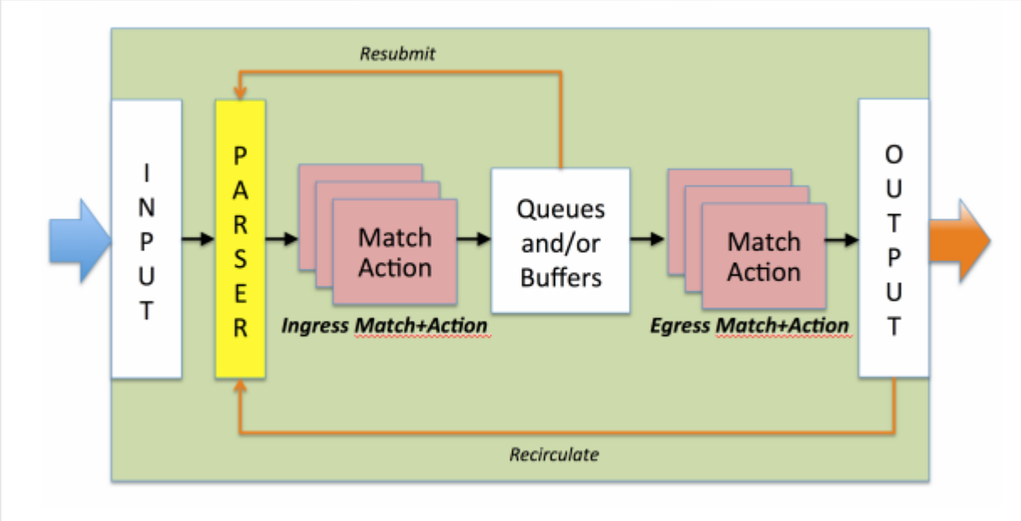


Multiple switching SW options, develop apps not NOS

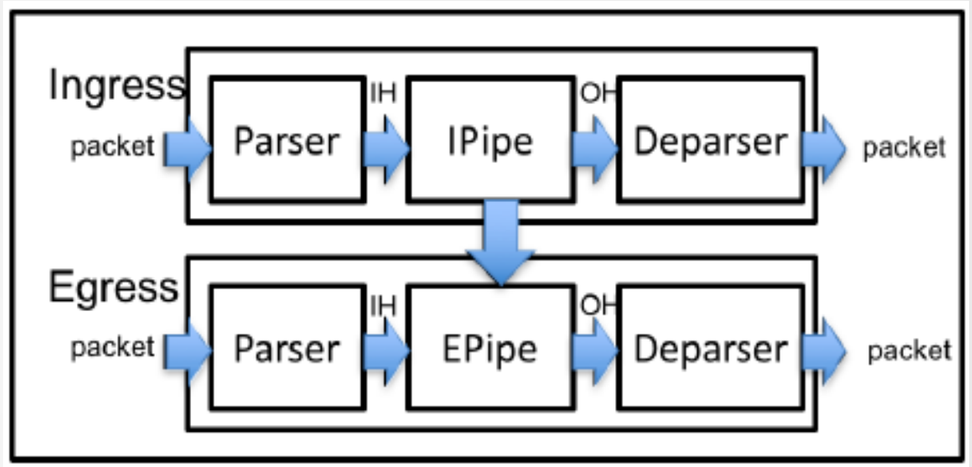
SAIFlexAPI – uniform API for all programing language

P4-16

v14



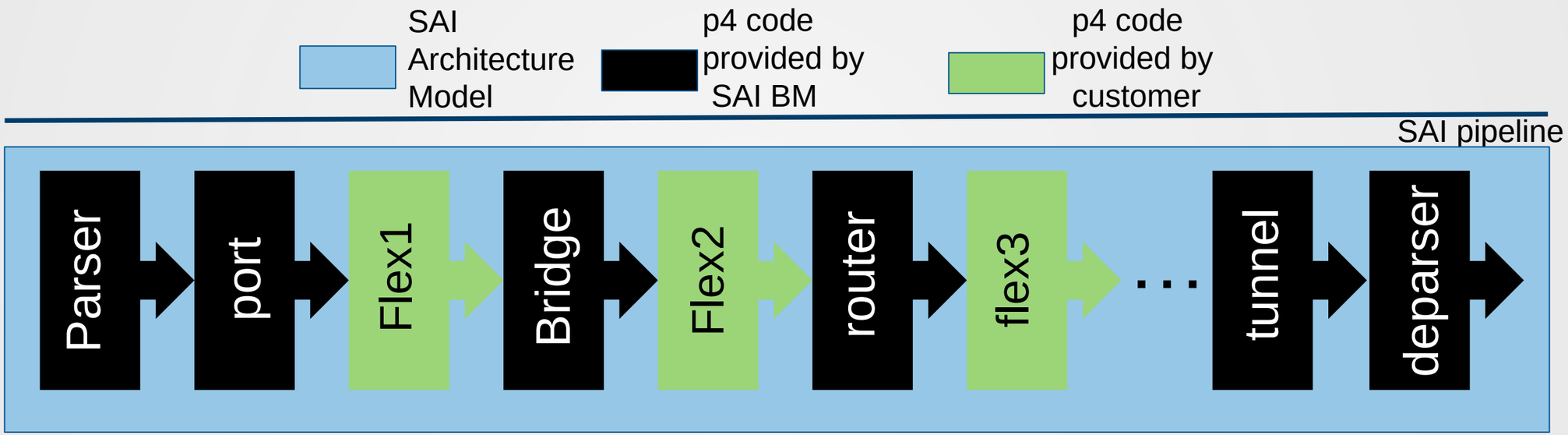
v16



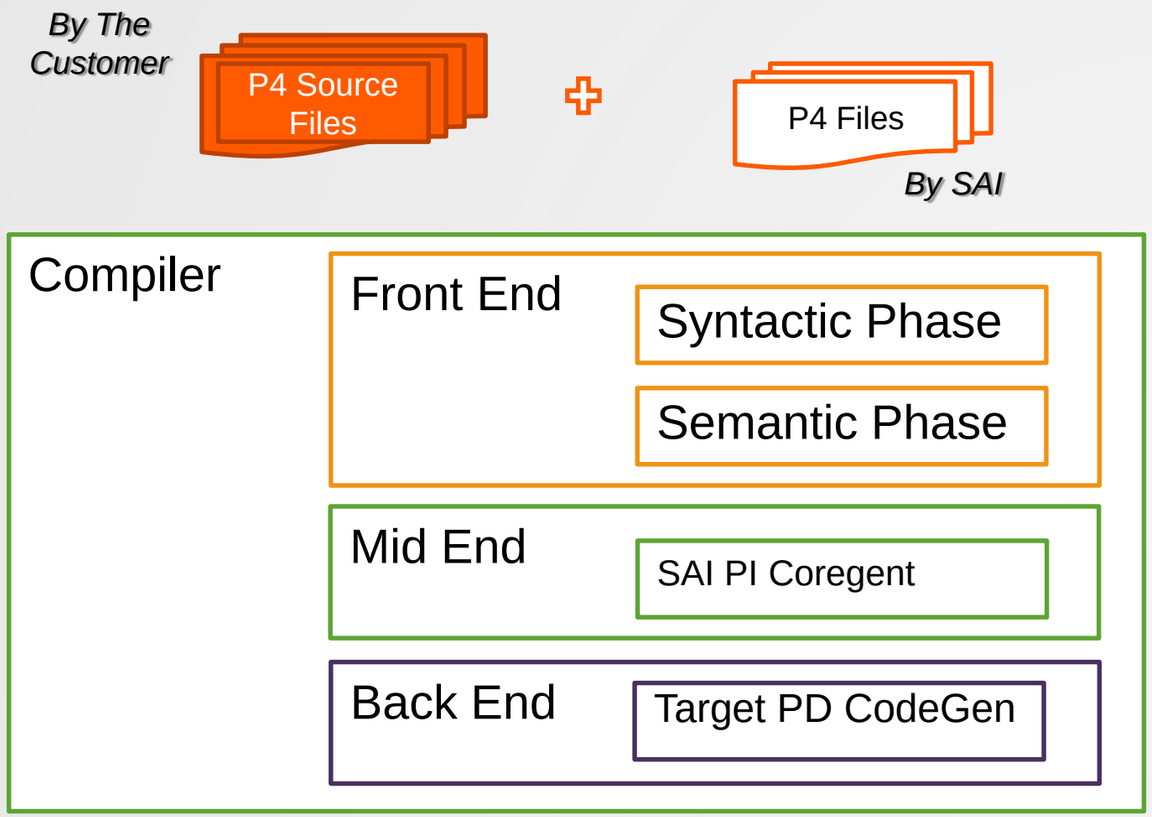
From the spec:

- Introducing P4 architecture description language
- “The P4 architecture can be thought of as a contract between the program and the target”
- “ Programmable blocks” i,.e. flexible blocks within a solid target
- “In general, P4 programs are not expected to be portable across different architectures”

SAI P4 model


SAI Architecture Model
p4 code provided by SAI BM
p4 code provided by customer
SAI pipeline
Parser
port
Flex1
Bridge
Flex2
router
flex3
. . .
tunnel
deparser

SAI P4 Compiler Architecture –HW



- **P4 version - $P4_{16}$,**
- **Front End** – Relates only to the language.
  - **Syntactic phase** – BNF based. Last time we read the source files. Output: Symbol tables.
  - **Semantic phase** – Verifies the Symbol tables. Extends the default semantic checks with platform specific ones.
- **Mid End**
  - **SAI Code gen (PI) generate SAI objects**
- **Back End**
  - **target code gen (PD) – vendor specific**
  - **SAI p4 switch(Soft Switch)**

Adding Bare Metal services to the Cloud

Goal

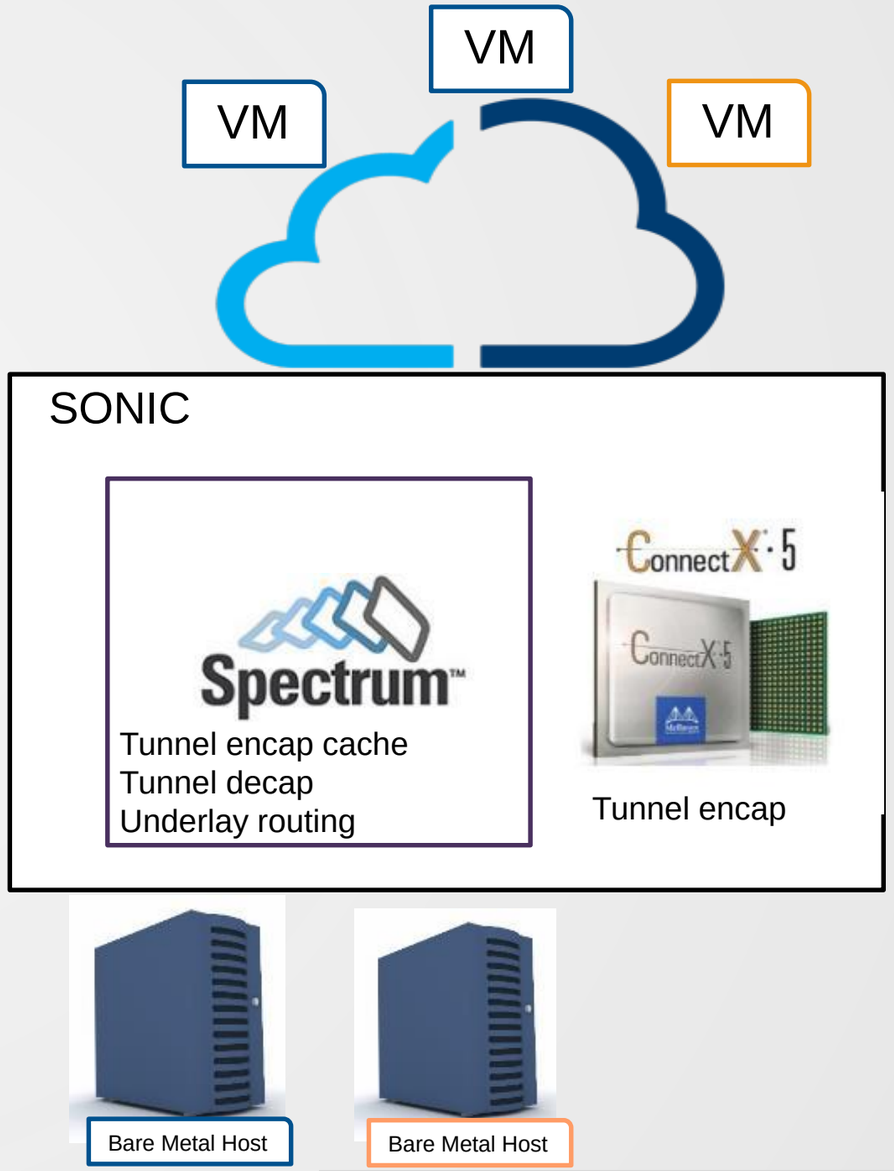
- Connect Bare Metal machine to cloud VMs

Challenges

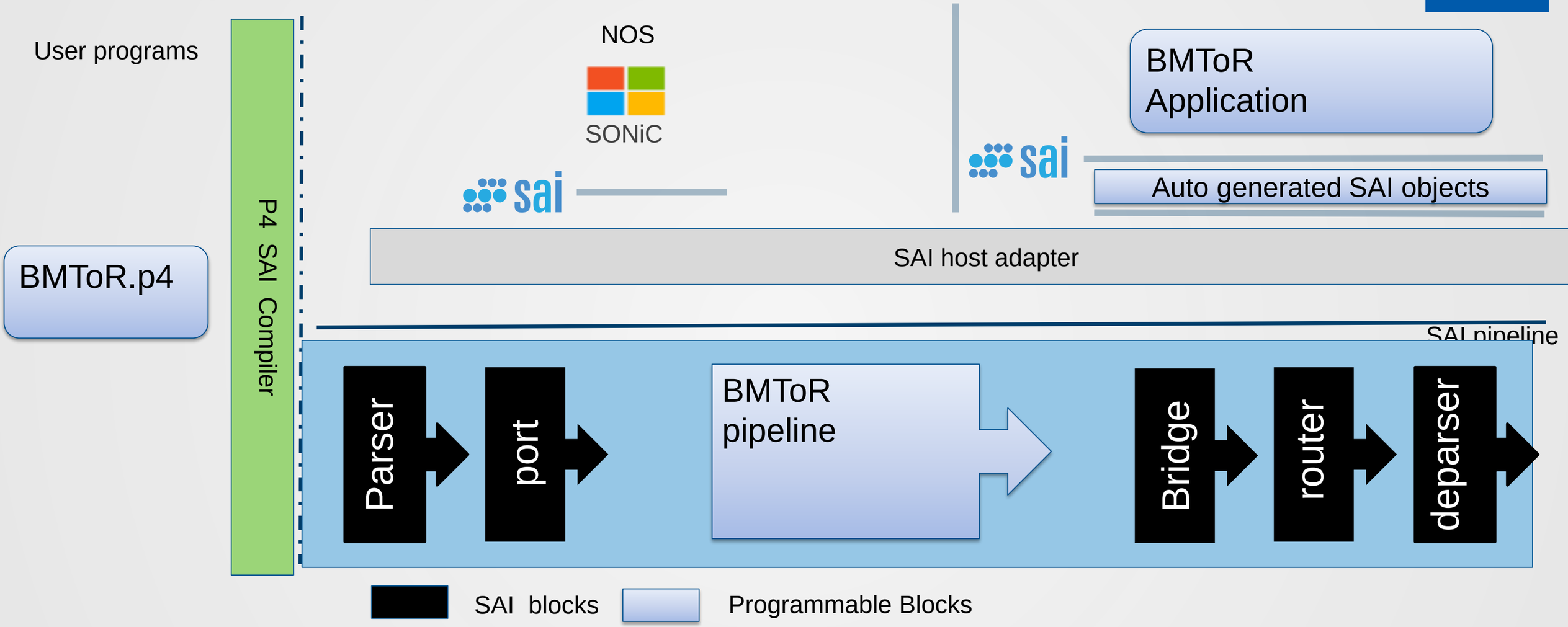
- Non standard encapsulation logic
- 10M tunnels

Solution

- Programmable pipeline implementation for encapsulation logic across Mellanox ICs
  - Uniform APIs (SAI) for the ConnectX-5 eSWITCH and Spectrum switch
  - On top of legacy switching features
- Host/ Switch integrated pipe
  - Switch role: cache for active recent flows
  - Host role - scale



SAI Programmability use case



Multiple switching SW options, develop apps not NOS
SAIFlexAPI – uniform API for all programing language

P4 Program

Auto generated JSON file

```
control control_in_port(inout Headers_t headers, inout metadata_t meta, inout standard_metadata_t
standard_metadata){
    #include "../inc/actions.p4"

    action set_vnet_bitmap(bit<12> vnet_bitmap){
        set_meta_reg(vnet_bitmap,0x0fff);
        hit_counter();
    }

    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);
        hit_counter();
    }

    table table_peering{
        key = {
            meta.metadatakeys.METADATA_SRC_PORT :exact;
            // TODO add vrf
        }
        actions = {set_vnet_bitmap;}
        size = PORTNUM;
    }

    table table_vhost{
        key = {
            meta.metadatakeys.METADATA_REG : ternary;
            headers.ip.v4.dstAddr :exact;
        }
        actions = {to_tunnel;
                   to_router;
                   to_port;}
        size=MSEE_TABLE_SIZE;
    }

    apply{
        table_peering.apply();
        table_vhost.apply();
                                                                    39,1          33%
```

```
yonatanp@yonatanp-VirtualBox:~/p4_16/flextrum$ p4c-mlnx-spc p4src/bm_tor/bm_tor.p4 -o bmtor.
json
Spectrum backend
```