PINS: P4 Integrated Network Stack
**PINS = SONiC + SDN**

- **SONiC** is widely deployed, modular, open source, and vendor agnostic
  - Runs a traditional control plane (e.g. BGP)
  - Solid foundation for SDN-enabled switch OS

- **Software-Defined Network (SDN)** brings many features and capabilities to the network which is difficult to achieve using traditional embedded control planes

- **P4 Integrated Network Stack (PINS)** adds SDN capabilities to SONiC:
  - *Formal Pipeline Specification: P4* used to model the SAI pipeline
  - *Remote API: P4Runtime* used to program objects into the pipeline
SONiC for PINS

SONiC is adapted to work with remote SDN controllers by adding the support for P4Runtime protocol

- Optional: Gives network operators a choice to incrementally migrate towards an SDN solution
- Switch Abstraction Interface (SAI) pipeline is modelled in P4 (SAI.p4)
- PINS translates between P4 entities and SAI API calls
- A response path is used to immediately inform the controller when an update succeeds or fails, compared to the fire-and-forget approach of other SONiC apps
- Remote packet I/O capability added for the SDN controller
SAI P4 Routing

sai/routing.p4

... @p4runtime_role(P4RUNTIME_ROLE_ROUTING)
@id(ROUTING_IPV4_TABLE_ID)
table ipv4_table {
  key = {
    // Sets vrf_id in sai_route_entry_t.
    local_metadata.vrf_id : exact @id(1) @name("vrf_id")
    @refers_to((vrf_table, vrf_id));
    // Sets destination in sai_route_entry_t to an IPv4 prefix.
    headers.ipv4.dst_addr : lpm @format(IPV4_ADDRESS) @id(2)
    @name("ipv4_dst");
  }
  actions = {
    @proto_id(1) drop;
    @proto_id(2) set_nexthop_id;
    @proto_id(3) set_wcmp_group_id;
    const default_action = drop;
    size = ROUTING_IPV4_TABLE_MINIMUM_GUARANTEED_SIZE;
  }
}
...

IP Routing (sairoute.h)

Match Keys:
- VRF ID
- IP Dest
Action (one of):
- Set Next Hop Group ID
- Set Next Hop ID

Next Hop Group (sainexthopgroup.h)

Match Keys:
- Next Hop Group ID
Action:
- Set Next Hop ID via WCMP

Next Hop (sainexthop.h)

Match Keys:
- Next Hop ID
Action:
- Set RIF ID
- Set Neighbor IP

Neighbor (saineighbor.h)

Match Keys:
- RIF ID
- Neighbor IP
Action:
- Set Dest MAC

Router Interface (sairouterinterface.h)

Match Keys:
- RIF ID
Action:
- Set Dest Port
- Set Src MAC

Legend:
P4 Table (maps to SAI header)
Users can define custom ACLs in P4
- Match fields
- Actions
- Counter
- Meters
- Table Size

P4 fields mapped to SAI using annotations

ACL tables are configured on the switch when the P4 pipeline is pushed via P4Runtime

SAI P4 ACLs

```
sai/custom/acl_ingress.p4

... table acl_ingress_table {
    key = {
        headers.ipv4.isValid() || headers.ipv6.isValid() : optional
        headers.ipv4.isValid() : optional
        headers.ipv6.isValid() : optional
        headers.ethernet.ether_type : ternary
        headers.ethernet.dst_addr : ternary
        headers.ipv4.dst_addr : ternary
        headers.ipv6.src_addr : ternary
        headers.ipv6.dst_addr : ternary
        ttl : ternary
        dscp : ternary
        ecn : ternary
        ip_protocol : ternary
        headers.icmp.type : ternary
        local_metadata.l4_dst_port : ternary
        local_metadata.ingress_port : optional
        @sai_field(SAI_ACL_TABLE_ATTR_FIELD_IP_TYPE);
        headers.ipv4.src_addr : ternary
        @sai_field(SAI_ACL_TABLE_ATTR_FIELD_IP_VERSION);
        headers.ipv6.src_addr : ternary
        @sai_field(SAI_ACL_TABLE_ATTR_FIELD_IP_VERSION);
        headers.ipv4.src_addr : ternary
        headers.ipv6.src_addr : ternary
        @sai_field(SAI_ACL_TABLE_ATTR_FIELD_SRC_IPV6);
        headers.ipv4.dst_addr : ternary
        @sai_field(SAI_ACL_TABLE_ATTR_FIELD_DST_IPV6);
        headers.ipv6.dst_addr : ternary
        @sai_field(SAI_ACL_TABLE_ATTR_FIELD_DST_IPV6);
        local_metadata.l4_src_port : ternary
        ...@sai_field(SAI_ACL_TABLE_ATTR_FIELD_L4_SRC_PORT);
        ...}
    actions = {
        copy();
        trap();
        forward();
        acl_drop(standard_metadata);
        @defaultonly NoAction;
    }
    const default_action = NoAction;
    meters = acl_ingress_meter;
    counters = acl_ingress_counter;
    size = 128;
}
```
PINS Topics

Use Cases: Weighted Cost MultiPath (WCMP)

SAI Generic Path Extension

PINS for Marvell
Weighted Cost MultiPath (WCMP)

WCMP distributes traffic flows on multiple links proportionally to the assigned weights

- Helps optimally distribute traffic in unbalanced networks
- Watch Google’s presentation on WCMP for more information here:

We implement WCMP using the open source **ONOS** and **SD-Fabric** platforms.

http://www.youtube.com/watch?v=HL9msGD4OMU
End-to-End Architecture

SD-Fabric is a P4 programmable network fabric that includes:

- **SDN Controller**
  - ONOS
  - sdfabric-control application

- **Leaf-spine fabric of programmable switches**
  - P4 Integrated Network Stack (PINS)
    - sai.p4
    - P4 entities to SAI APIs call
    - Remote Packet I/O
**WCMP vs ECMP**

**ECMP Weight Distribution Example**

- ECMP equally distributes the traffic through the links
- Not optimal with unbalanced networks

---

![ECMP Weight Distribution Example](image-url)
WCMP vs ECMP

WCMP Weight Distribution Example

- WCMP distributes the traffic through the links according to their weight.
- Can reach optimality with unbalanced networks.

![Diagram showing WCMP weight distribution example]
**WCMP vs ECMP**

**WCMP Weight Distribution Example**

- WCMP distributes the traffic through the links according to their **weight**
- Can reach optimality with unbalanced networks
WCMP vs ECMP
WCMP Weight Distribution Example

- WCMP distributes the traffic through the links according to their weight
- Can reach optimality with unbalanced networks
WCMP Demo Using PINS and ONOS
PINS Generic SAI Extensions

OCP Demo
Faster development/deployment cycle for features extension to SAI pipeline
Uniform P4Runtime abstraction
Solution enables SDN managed
- SAI tables
- SAI Extension tables
Field upgradeable and extensible
Brings agility and differentiation with specialized use cases
Customer-specific network headers and data plane functions
APPL_DB:
"P4RT_TABLE:EXT_QINQ_TABLE:\"match/inner_vlan\":\"0x07b\",
\"match/outer_vlan\":\"0x12c\"\""
1) "action"
2) "set_subscriber_id"
3) "param/subscriber_id"
4) "0x000007d0"

Attributes in ASIC_DB:

SAI_GENERIC_PROGRAMMABLE_ATTR_OBJECT_NAME: qinq_table
SAI_GENERIC_PROGRAMMABLE_ATTR_ENTRY:
[{
"inner_vlan":{"datatype":"SAI_ATTR_VALUE_TYPE_UINT16","value":"0x07b"}
},{
"outer_vlan":{"datatype":"SAI_ATTR_VALUE_TYPE_UINT16","value":"0x12c"}
},{
"set_subscriber_id":{"subscriber_id":{"datatype":"SAI_ATTR_VALUE_TYPE_UINT32","value":"0x000007d0"}}
}]
This demo shows:

- New table definition extending SAI.p4
- Use of P4RT shell as P4RT Client
  - Connects to P4RT Server running SONiC
  - Triggers creation/removal of extension table entries
- SONiC APPL_DB
  - P4RT Tables definitions in APPL_DB
  - Addition/removal of extended table entries in APPL_DB
- Generic SAI Extension API parameters

Details at these stages shown in the demo

Work in Progress
- Extension APIs approved and currently in process of integration with SONiC
- SONiC control-plane design reviewed with community
PINS Infrastructure

This repository contains infrastructure and libraries that assist in testing P4 Integrated Network Stack (PINS) switches.
Marvell has implemented PINS on their Teralynx® devices

- Supporting SONiC 202111 and SAI 1.9
- Features include:
  - ACLS
  - WCMP
  - Routing
  - Mirroring
  - CPU Path RX/TX
  - Fast Reset (sub 100ms)

Visit the Marvell booth for more details.
Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.
No product or component can be absolutely secure.
Your costs and results may vary.
© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
Thank you!
Questions?