DASH Flow API

Zhixiong Niu, Clark Lee, Feng Yan

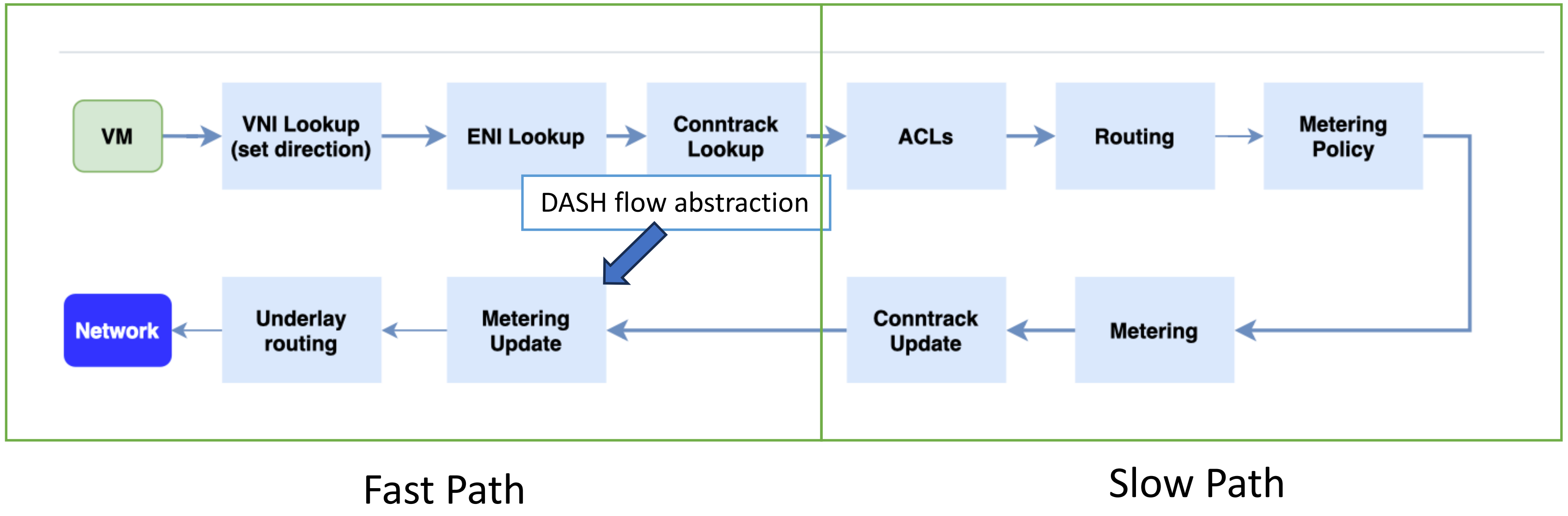
April 2024

Outline

- Introduction
- Use Cases
    - Use case #1: Network Gateway
    - Use case #2: SmartSwitch HA
    - Use case #3: Flow Diagnosis
- DASH Flow Design
- Implement use cases with DASH Flow API

Introduction

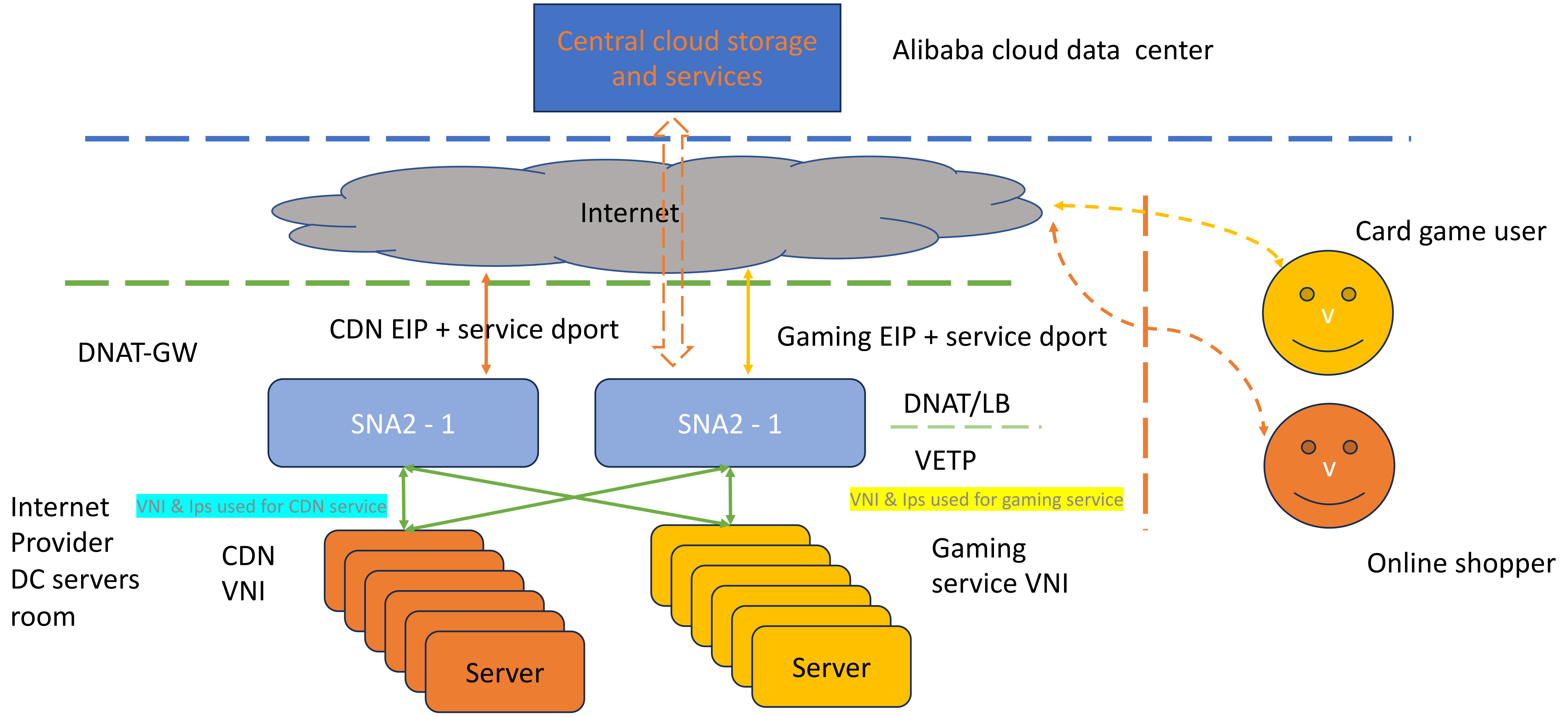
- DASH supports the storage and processing of millions of flow states.
- To further enhance the DASH flow processing capabilities, we offer a **DASH flow abstraction** layer to facilitate vendor-neutral flow management.



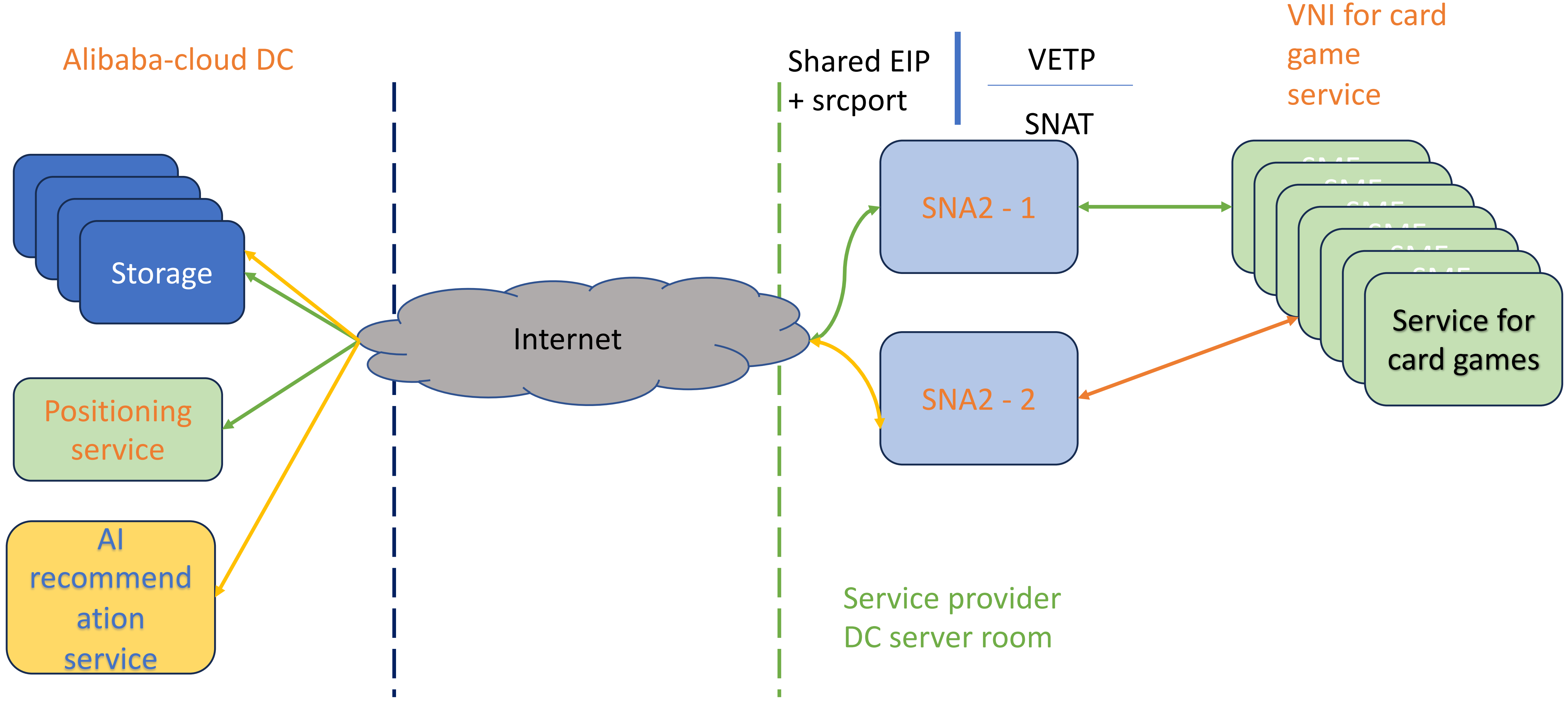
NAT Gateway in Alibaba

Use Case

NAT Gateway in Alibaba – DNAT/LB


Central cloud storage and services
Alibaba cloud data center
Internet
Card game user
DNAT-GW
CDN EIP + service dport
Gaming EIP + service dport
SNA2 - 1
SNA2 - 1
DNAT/LB
VETP
Internet Provider DC servers room
VNI & Ips used for CDN service
VNI & Ips used for gaming service
CDN VNI
Gaming service VNI
Online shopper
Server
Server

NAT Gateway in Alibaba - SNAT

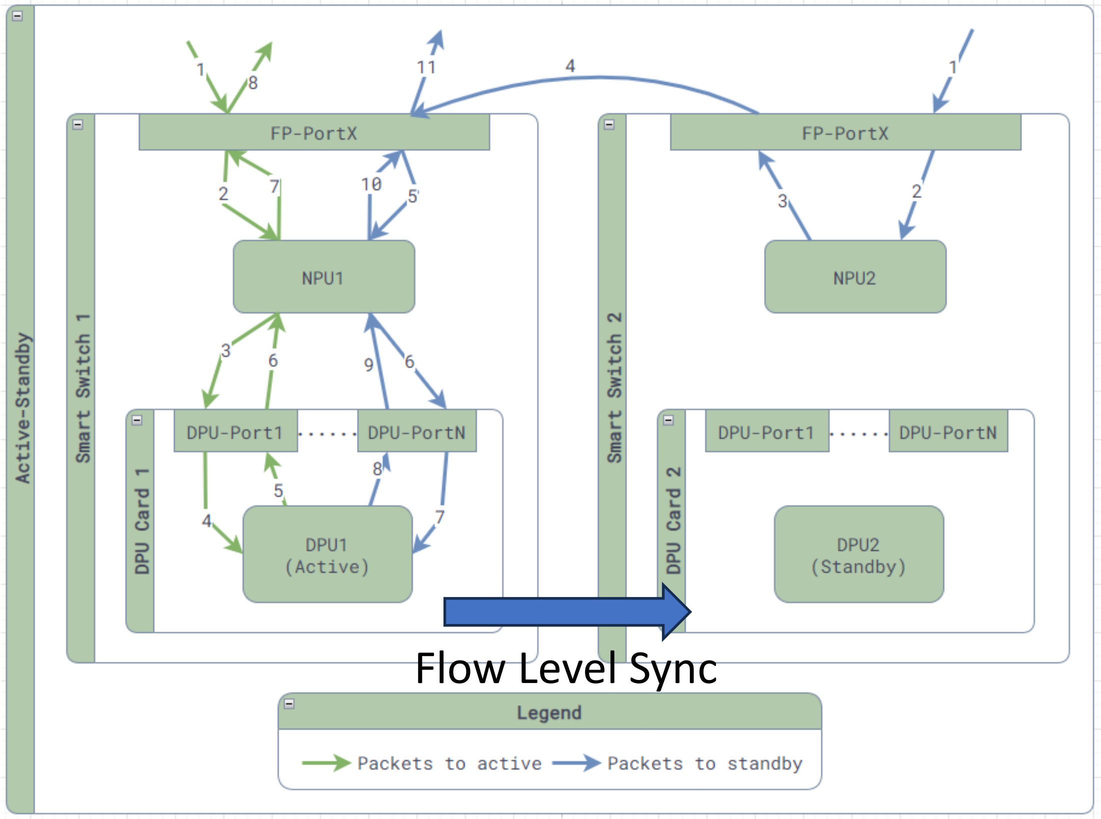


Use Cases in MSFT

Use Cases

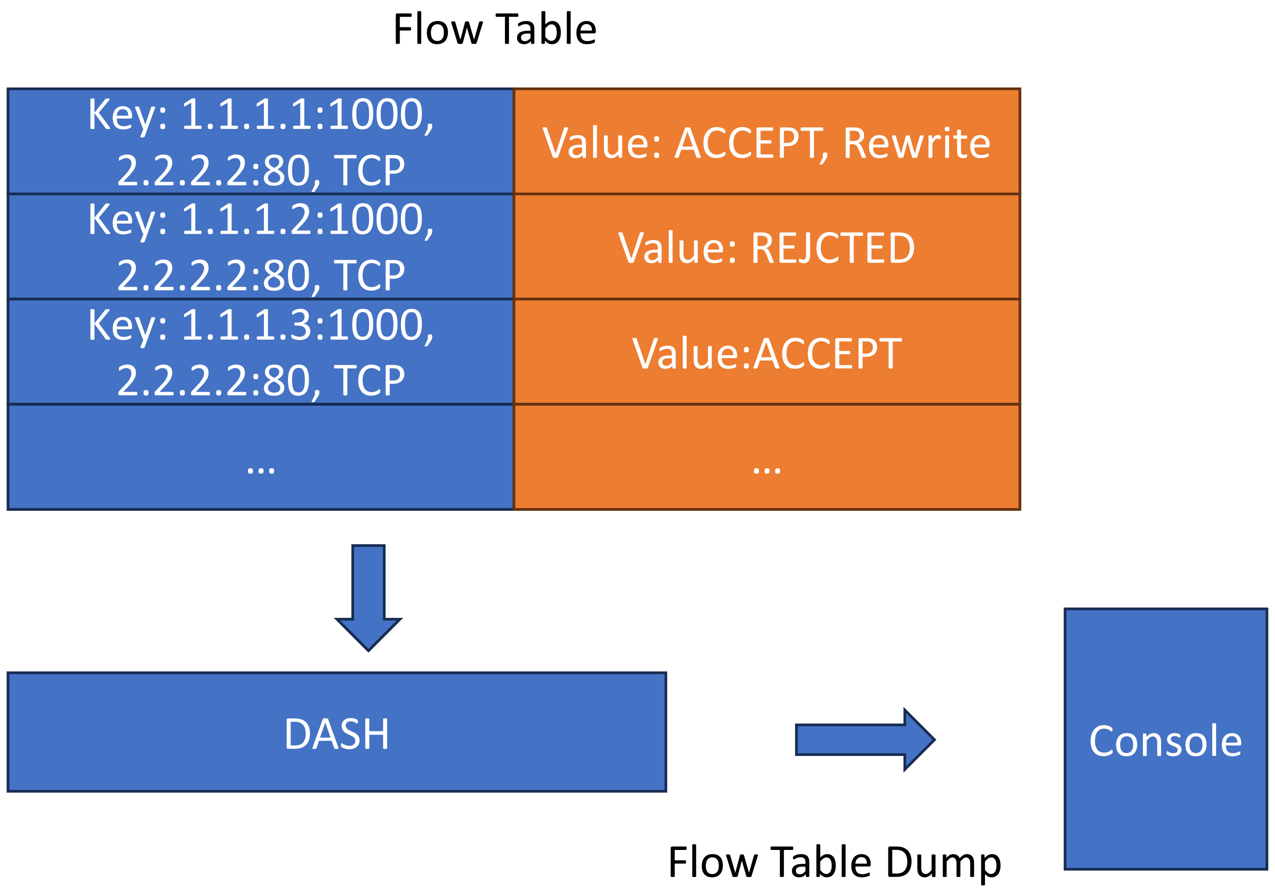
Use Case #1: SmartSwitch HA

- Smart Switch HA
  - A single switch / DPU goes down or when a network failure happens, it will not cause existing flows to be dropped.
- Requirement
  - Flow Level Sync
    - Get flow entries from Active DPU
    - Create/update entries to Standby DPU



Use Case #2: Flow Diagnosis

- Flow Diagnosis
  - E.g. VM A to VM B is not working
    - Diagnosis: get sample flow entries from VM A to VM B and check the flow entries (flow key and flow state) from console or other ways


Flow Table

Key: 1.1.1.1:1000, 2.2.2.2:80, TCP	Value: ACCEPT, Rewrite
Key: 1.1.1.2:1000, 2.2.2.2:80, TCP	Value: REJCTED
Key: 1.1.1.3:1000, 2.2.2.2:80, TCP	Value:ACCEPT
…	…

DASH Flow Design

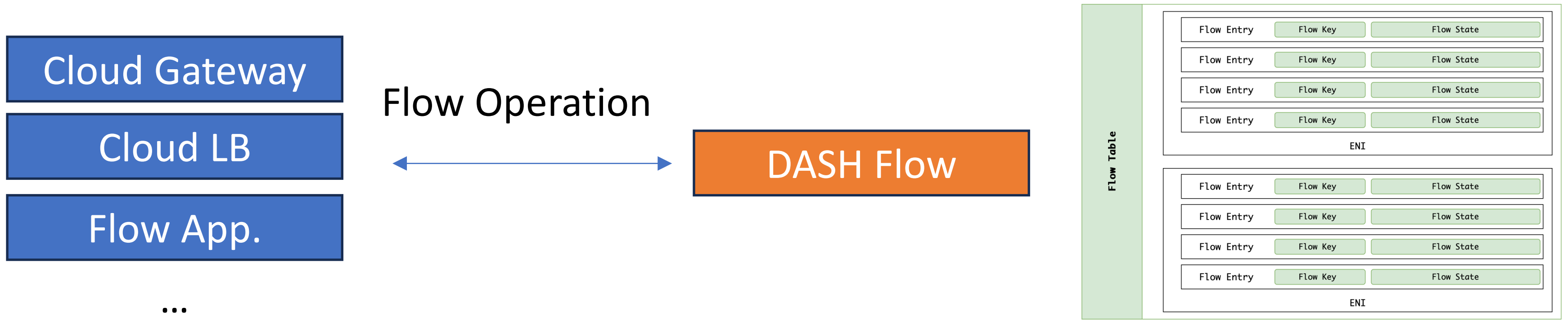
https://github.com/sonic-net/DASH/pull/462

DASH Flow Design: Intro

- Goals
  - Supports storage and processing of millions of flow states
  - Enhances DASH flow processing capabilities through a vendor-neutral management layer
- Key features
  - Offers a unified control dash flow interface across dash devices.
  - Provides abstraction of flow table and flow entries.
  - Includes APIs for comprehensive flow management (creation, removal, retrieval, configuration).

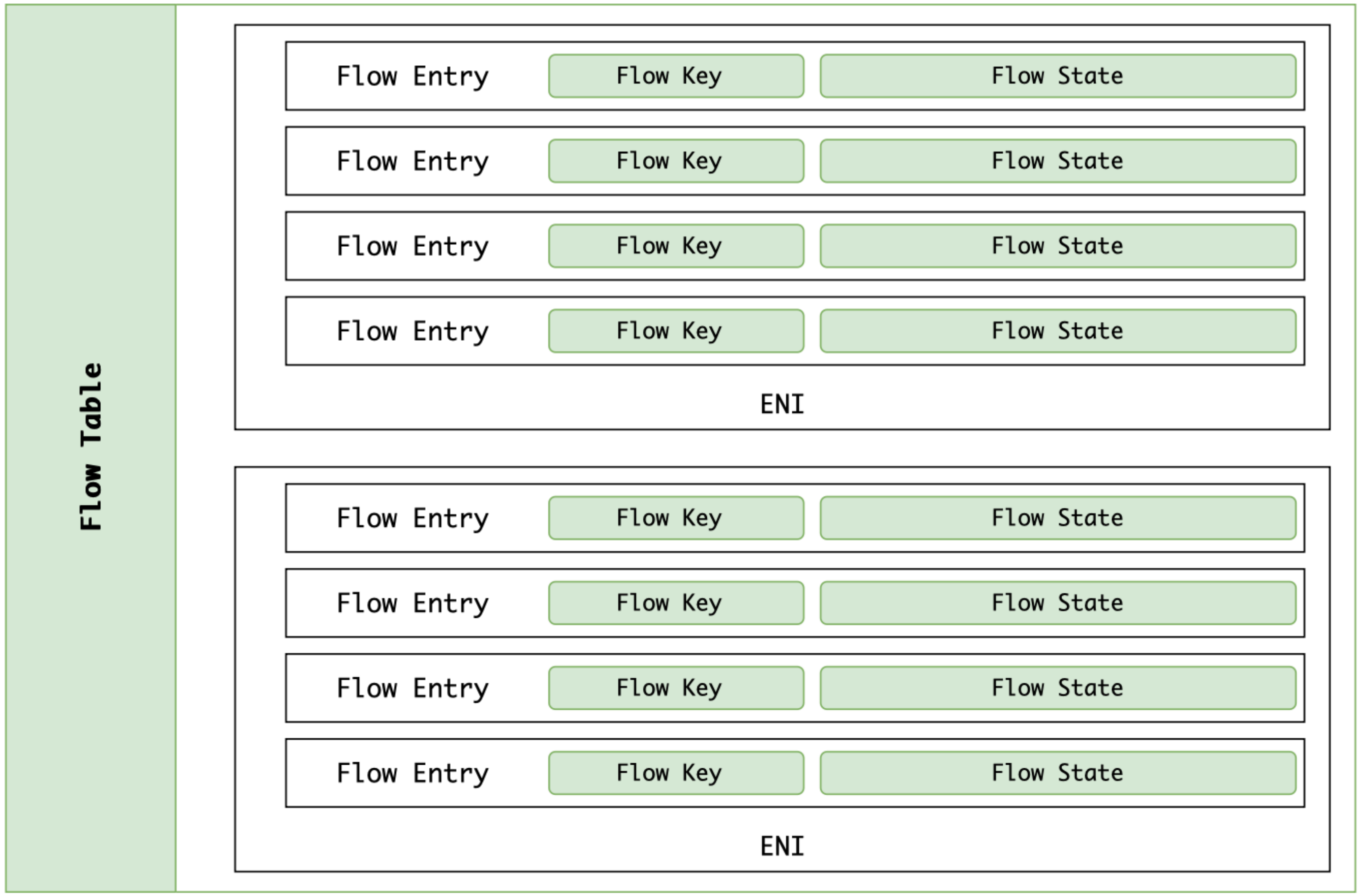
DASH Flow Design: Benefits and Scenarios

- Benefits
  - Enhanced control and flexibility in managing flows in DASH
  - Ability to develop services for diverse scenarios
- Supported Scenarios
  - Dataplane Applications: Cloud gateways, load balancers.
  - Flow Management: Flow offloading, updating, redirection, re-simulation.
  - Dataplane Debugging: Diagnosing flow behaviors.
  - Foundational Flow Services: Ensuring flow state high availability.



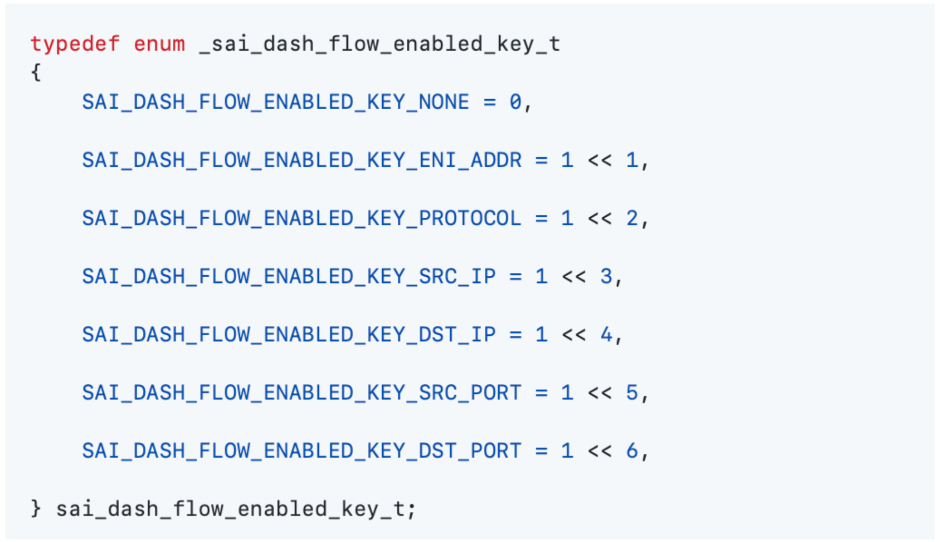
DASH Flow Design: Abstraction

- Flow Table
  - Create, Remove, Set, Get
- Flow Entry
  - Create, Remove, Set, Get
  - Bulk Get Session with filters



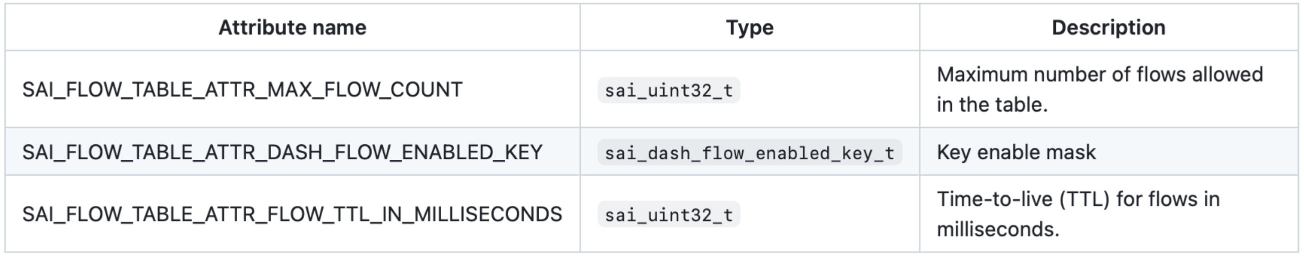
DASH Flow Design – Flow Table

- Defined as sai_objects
- Support different enabled keys
- Support TTL for flow entries



```
typedef enum _sai_dash_flow_enabled_key_t
{
    SAI_DASH_FLOW_ENABLED_KEY_NONE = 0,

    SAI_DASH_FLOW_ENABLED_KEY_ENI_ADDR = 1 << 1,

    SAI_DASH_FLOW_ENABLED_KEY_PROTOCOL = 1 << 2,

    SAI_DASH_FLOW_ENABLED_KEY_SRC_IP = 1 << 3,

    SAI_DASH_FLOW_ENABLED_KEY_DST_IP = 1 << 4,

    SAI_DASH_FLOW_ENABLED_KEY_SRC_PORT = 1 << 5,

    SAI_DASH_FLOW_ENABLED_KEY_DST_PORT = 1 << 6,

} sai_dash_flow_enabled_key_t;
```

Enabled key bitmap

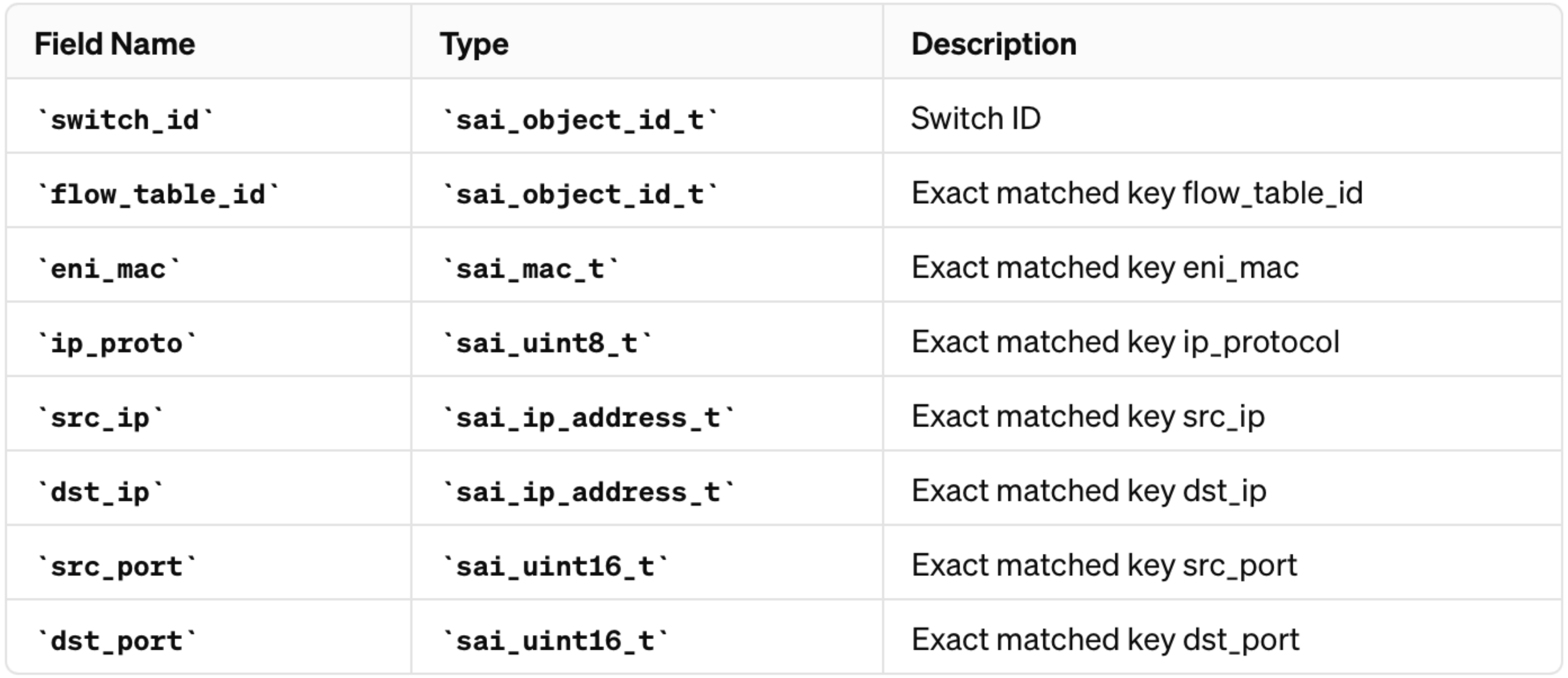


Attribute name	Type	Description
SAI_FLOW_TABLE_ATTR_MAX_FLOW_COUNT	`sai_uint32_t`	Maximum number of flows allowed in the table.
SAI_FLOW_TABLE_ATTR_DASH_FLOW_ENABLED_KEY	`sai_dash_flow_enabled_key_t`	Key enable mask
SAI_FLOW_TABLE_ATTR_FLOW_TTL_IN_MILLISECONDS	`sai_uint32_t`	Time-to-live (TTL) for flows in milliseconds.

Attributes of the flow table

DASH Flow Design – Flow Entry

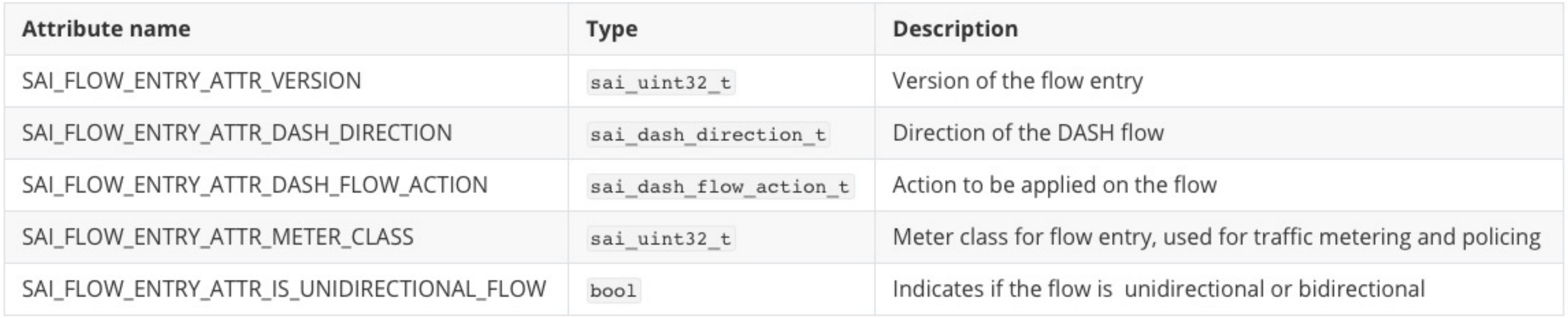
- Keys



Field Name	Type	Description
`switch_id`	`sai_object_id_t`	Switch ID
`flow_table_id`	`sai_object_id_t`	Exact matched key flow_table_id
`eni_mac`	`sai_mac_t`	Exact matched key eni_mac
`ip_proto`	`sai_uint8_t`	Exact matched key ip_protocol
`src_ip`	`sai_ip_address_t`	Exact matched key src_ip
`dst_ip`	`sai_ip_address_t`	Exact matched key dst_ip
`src_port`	`sai_uint16_t`	Exact matched key src_port
`dst_port`	`sai_uint16_t`	Exact matched key dst_port

DASH Flow Design – Flow Entry Attributes

- Flow basic metadata

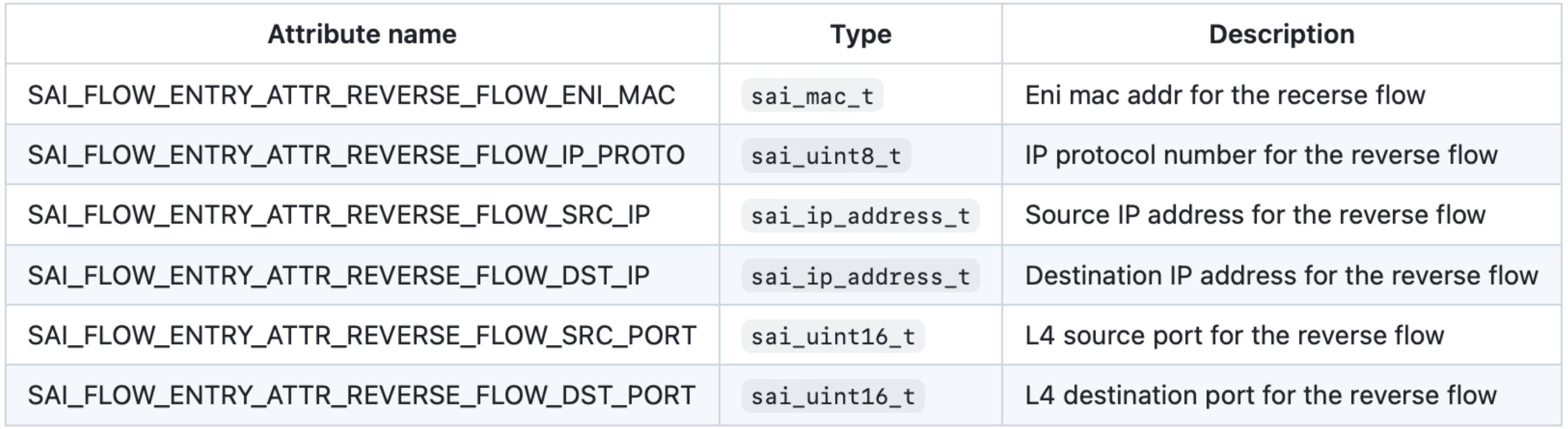


Attribute name	Type	Description
SAI_FLOW_ENTRY_ATTR_VERSION	`sai_uint32_t`	Version of the flow entry
SAI_FLOW_ENTRY_ATTR_DASH_DIRECTION	`sai_dash_direction_t`	Direction of the DASH flow
SAI_FLOW_ENTRY_ATTR_DASH_FLOW_ACTION	`sai_dash_flow_action_t`	Action to be applied on the flow
SAI_FLOW_ENTRY_ATTR_METER_CLASS	`sai_uint32_t`	Meter class for flow entry, used for traffic metering and policing
SAI_FLOW_ENTRY_ATTR_IS_UNIDIRECTIONAL_FLOW	`bool`	Indicates if the flow is unidirectional or bidirectional

Fig. basic metadata attributes

DASH Flow Design – Flow Entry Attributes

- Flow entry attributes for reverse flow

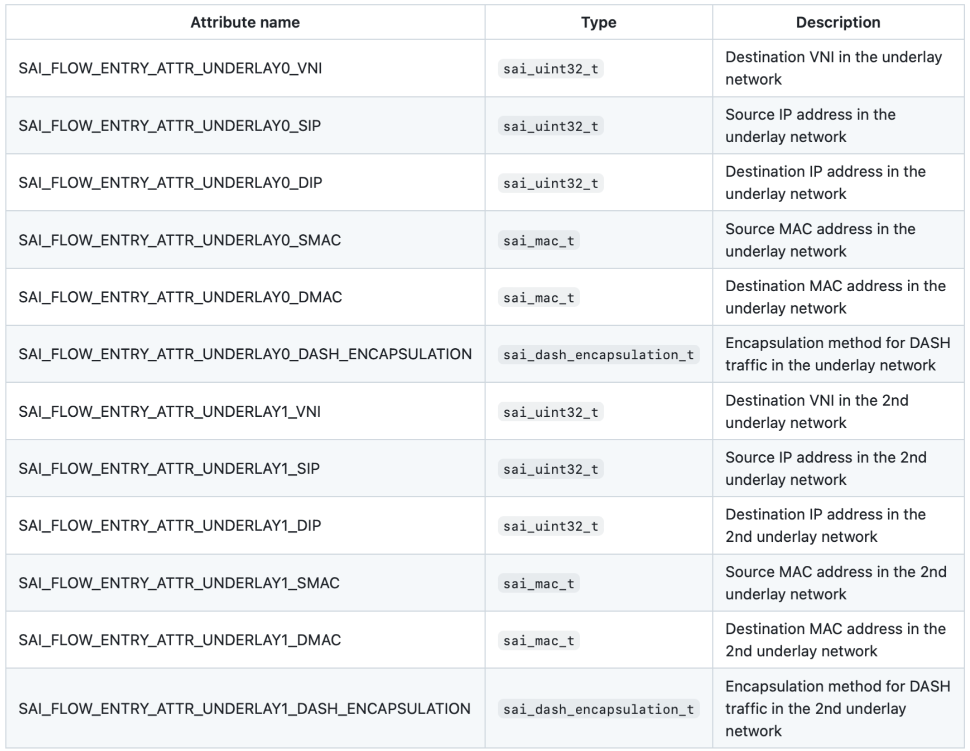


Attribute name	Type	Description
SAI_FLOW_ENTRY_ATTR_REVERSE_FLOW_ENI_MAC	`sai_mac_t`	Eni mac addr for the recerse flow
SAI_FLOW_ENTRY_ATTR_REVERSE_FLOW_IP_PROTO	`sai_uint8_t`	IP protocol number for the reverse flow
SAI_FLOW_ENTRY_ATTR_REVERSE_FLOW_SRC_IP	`sai_ip_address_t`	Source IP address for the reverse flow
SAI_FLOW_ENTRY_ATTR_REVERSE_FLOW_DST_IP	`sai_ip_address_t`	Destination IP address for the reverse flow
SAI_FLOW_ENTRY_ATTR_REVERSE_FLOW_SRC_PORT	`sai_uint16_t`	L4 source port for the reverse flow
SAI_FLOW_ENTRY_ATTR_REVERSE_FLOW_DST_PORT	`sai_uint16_t`	L4 destination port for the reverse flow

Fig. Reverse flow key attributes

DASH Flow Design – Flow Entry Attributes

- Flow encap related attributes
  - Support two layers of underlay

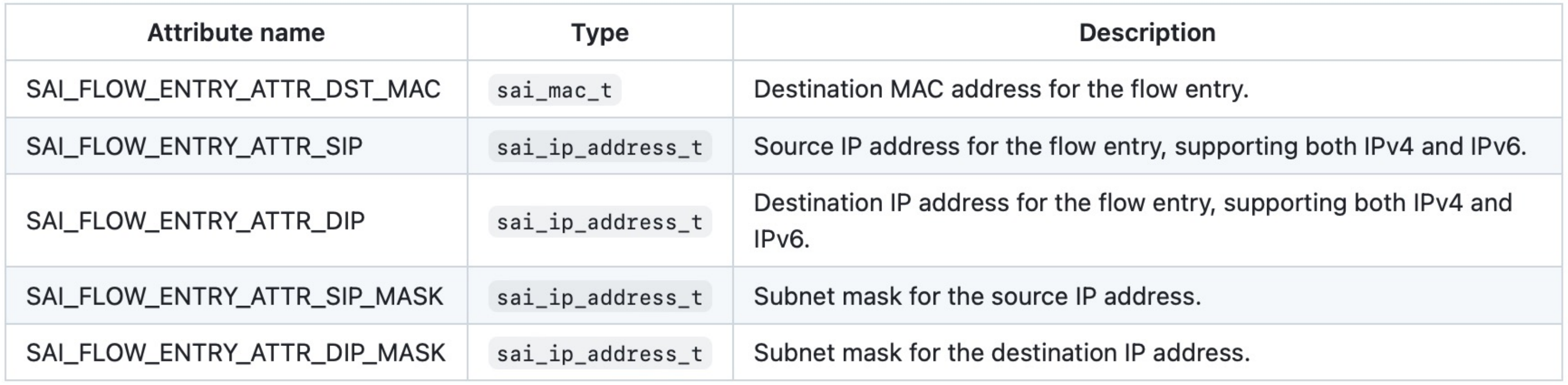


Attribute name	Type	Description
SAI_FLOW_ENTRY_ATTR_UNDERLAY0_VNI	`sai_uint32_t`	Destination VNI in the underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY0_SIP	`sai_uint32_t`	Source IP address in the underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY0_DIP	`sai_uint32_t`	Destination IP address in the underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY0_SMAC	`sai_mac_t`	Source MAC address in the underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY0_DMAC	`sai_mac_t`	Destination MAC address in the underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY0_DASH_ENCAPSULATION	`sai_dash_encapsulation_t`	Encapsulation method for DASH traffic in the underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY1_VNI	`sai_uint32_t`	Destination VNI in the 2nd underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY1_SIP	`sai_uint32_t`	Source IP address in the 2nd underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY1_DIP	`sai_uint32_t`	Destination IP address in the 2nd underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY1_SMAC	`sai_mac_t`	Source MAC address in the 2nd underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY1_DMAC	`sai_mac_t`	Destination MAC address in the 2nd underlay network
SAI_FLOW_ENTRY_ATTR_UNDERLAY1_DASH_ENCAPSULATION	`sai_dash_encapsulation_t`	Encapsulation method for DASH traffic in the 2nd underlay network

Fig. Flow encap related attributes

DASH Flow Design – Flow Entry Attributes

- Flow overlay rewrite related attributes

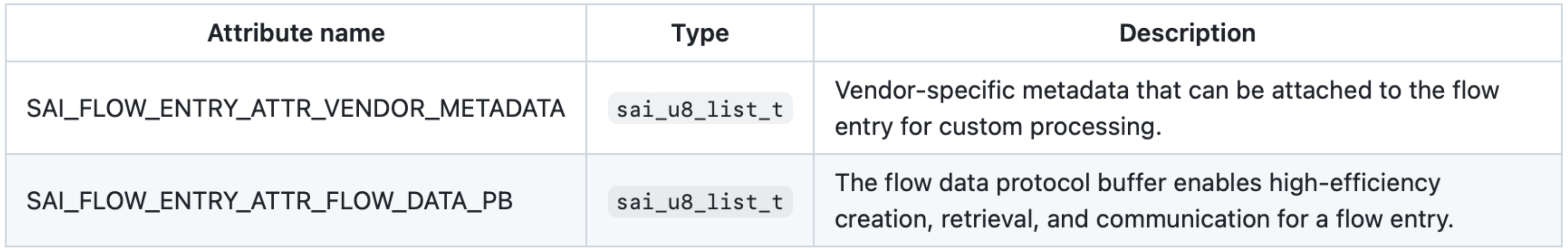


Attribute name	Type	Description
SAI_FLOW_ENTRY_ATTR_DST_MAC	`sai_mac_t`	Destination MAC address for the flow entry.
SAI_FLOW_ENTRY_ATTR_SIP	`sai_ip_address_t`	Source IP address for the flow entry, supporting both IPv4 and IPv6.
SAI_FLOW_ENTRY_ATTR_DIP	`sai_ip_address_t`	Destination IP address for the flow entry, supporting both IPv4 and IPv6.
SAI_FLOW_ENTRY_ATTR_SIP_MASK	`sai_ip_address_t`	Subnet mask for the source IP address.
SAI_FLOW_ENTRY_ATTR_DIP_MASK	`sai_ip_address_t`	Subnet mask for the destination IP address.

Fig. Flow overlay rewrite related attributes

DASH Flow Design – Flow Entry Attributes

- Extra flow metadata

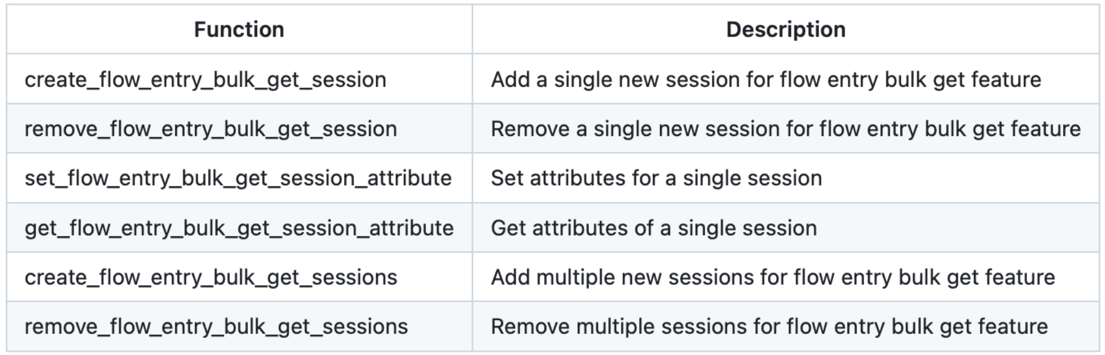


Attribute name	Type	Description
SAI_FLOW_ENTRY_ATTR_VENDOR_METADATA	`sai_u8_list_t`	Vendor-specific metadata that can be attached to the flow entry for custom processing.
SAI_FLOW_ENTRY_ATTR_FLOW_DATA_PB	`sai_u8_list_t`	The flow data protocol buffer enables high-efficiency creation, retrieval, and communication for a flow entry.

Fig. Extra flow metadata

DASH Flow Design – Flow Bulk Get Session

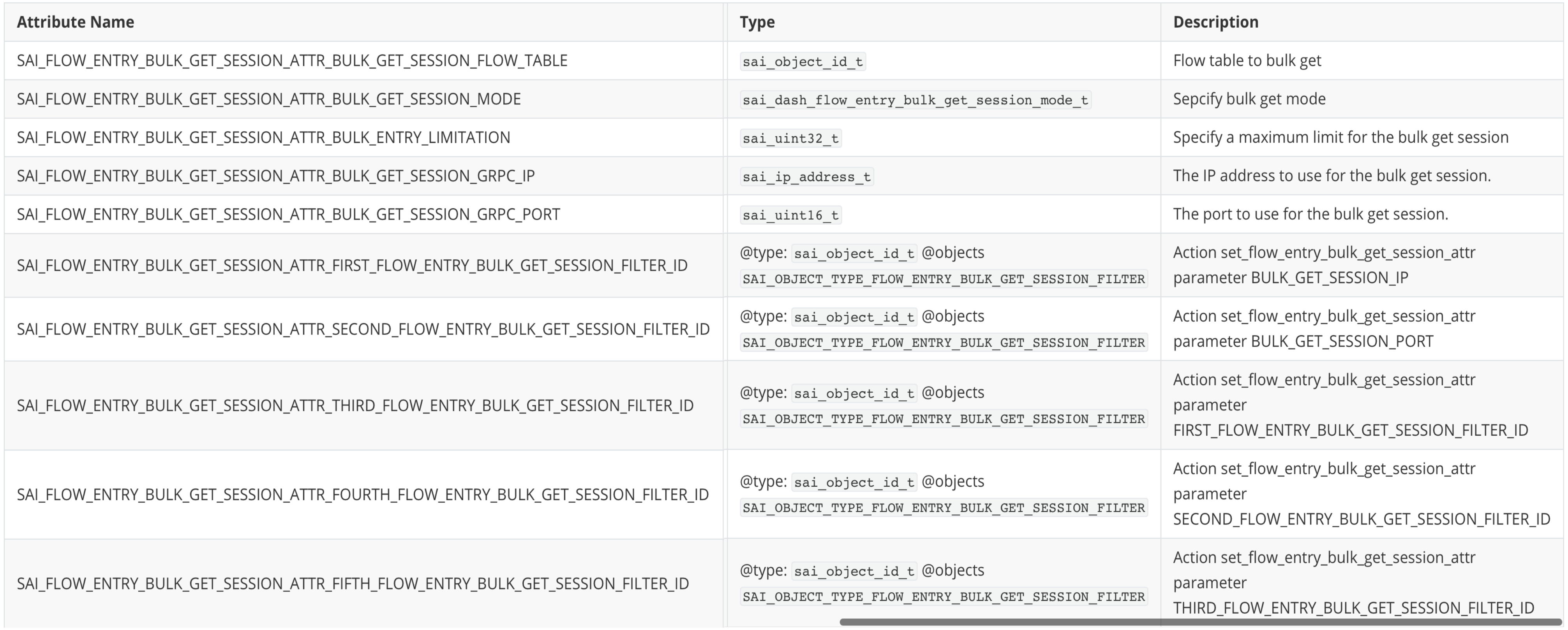
- Goal
  - Get/Set millions of flows which are changing all the time.
  - Allow dash provider optimize flow sync process based on their architecture
- Features
  - Two ways of receive flow entries
    - gRPC target server: Use a gRPC server to receive flows
    - Event notification: User can decide when to fetch flows
  - Support filters (up to five)



Function	Description
create_flow_entry_bulk_get_session	Add a single new session for flow entry bulk get feature
remove_flow_entry_bulk_get_session	Remove a single new session for flow entry bulk get feature
set_flow_entry_bulk_get_session_attribute	Set attributes for a single session
get_flow_entry_bulk_get_session_attribute	Get attributes of a single session
create_flow_entry_bulk_get_sessions	Add multiple new sessions for flow entry bulk get feature
remove_flow_entry_bulk_get_sessions	Remove multiple sessions for flow entry bulk get feature

Fig. Bulk Session APIs

DASH Flow Design – Flow Bulk Get Session

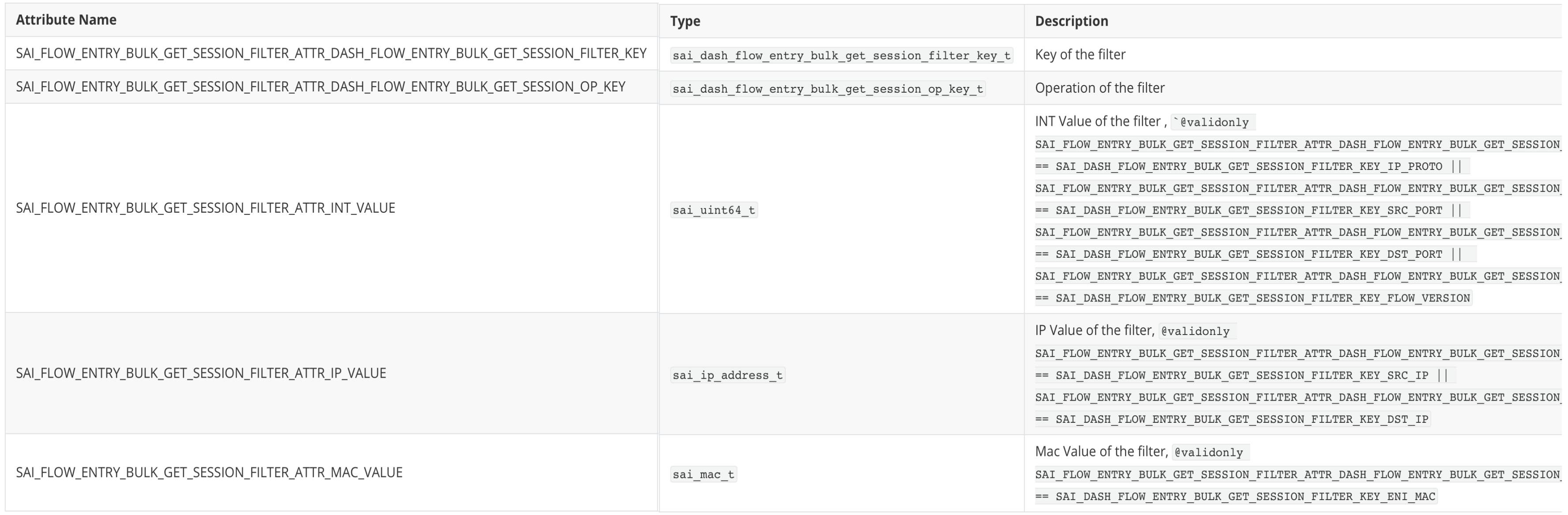


Attribute Name	Type	Description
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_BULK_GET_SESSION_FLOW_TABLE	`sai_object_id_t`	Flow table to bulk get
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_BULK_GET_SESSION_MODE	`sai_dash_flow_entry_bulk_get_session_mode_t`	Sepcify bulk get mode
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_BULK_ENTRY_LIMITATION	`sai_uint32_t`	Specify a maximum limit for the bulk get session
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_BULK_GET_SESSION_GRPC_IP	`sai_ip_address_t`	The IP address to use for the bulk get session.
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_BULK_GET_SESSION_GRPC_PORT	`sai_uint16_t`	The port to use for the bulk get session.
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_FIRST_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID	@type: `sai_object_id_t` @objects `SAI_OBJECT_TYPE_FLOW_ENTRY_BULK_GET_SESSION_FILTER`	Action set_flow_entry_bulk_get_session_attr parameter BULK_GET_SESSION_IP
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_SECOND_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID	@type: `sai_object_id_t` @objects `SAI_OBJECT_TYPE_FLOW_ENTRY_BULK_GET_SESSION_FILTER`	Action set_flow_entry_bulk_get_session_attr parameter BULK_GET_SESSION_PORT
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_THIRD_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID	@type: `sai_object_id_t` @objects `SAI_OBJECT_TYPE_FLOW_ENTRY_BULK_GET_SESSION_FILTER`	Action set_flow_entry_bulk_get_session_attr parameter FIRST_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_FOURTH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID	@type: `sai_object_id_t` @objects `SAI_OBJECT_TYPE_FLOW_ENTRY_BULK_GET_SESSION_FILTER`	Action set_flow_entry_bulk_get_session_attr parameter SECOND_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID
SAI_FLOW_ENTRY_BULK_GET_SESSION_ATTR_FIFTH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID	@type: `sai_object_id_t` @objects `SAI_OBJECT_TYPE_FLOW_ENTRY_BULK_GET_SESSION_FILTER`	Action set_flow_entry_bulk_get_session_attr parameter THIRD_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ID

Fig. Bulk Session Attributes

DASH Flow Design – Bulk Get Session Filter

- Filter is defined as sai_object#
  - Filter Key, Filter OP, Filter Value (INT, IP, MAC)
    - E.g. Prefect Sync: Sync all flow with version < 5
    - E.g. Diagnosis: Output 10 flows from VM A to VM B:443



Attribute Name	Type	Description
SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY	`sai_dash_flow_entry_bulk_get_session_filter_key_t`	Key of the filter
SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_OP_KEY	`sai_dash_flow_entry_bulk_get_session_op_key_t`	Operation of the filter
SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_INT_VALUE	`sai_uint64_t`	INT Value of the filter , `` `@validonly `` `SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_` `== SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_IP_PROTO ||` `SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_` `== SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_SRC_PORT ||` `SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_` `== SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_DST_PORT ||` `SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_` `== SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_FLOW_VERSION`
SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_IP_VALUE	`sai_ip_address_t`	IP Value of the filter, `@validonly` `SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_` `== SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_SRC_IP ||` `SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_` `== SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_DST_IP`
SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_MAC_VALUE	`sai_mac_t`	Mac Value of the filter, `@validonly` `SAI_FLOW_ENTRY_BULK_GET_SESSION_FILTER_ATTR_DASH_FLOW_ENTRY_BULK_GET_SESSION_` `== SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_ENI_MAC`

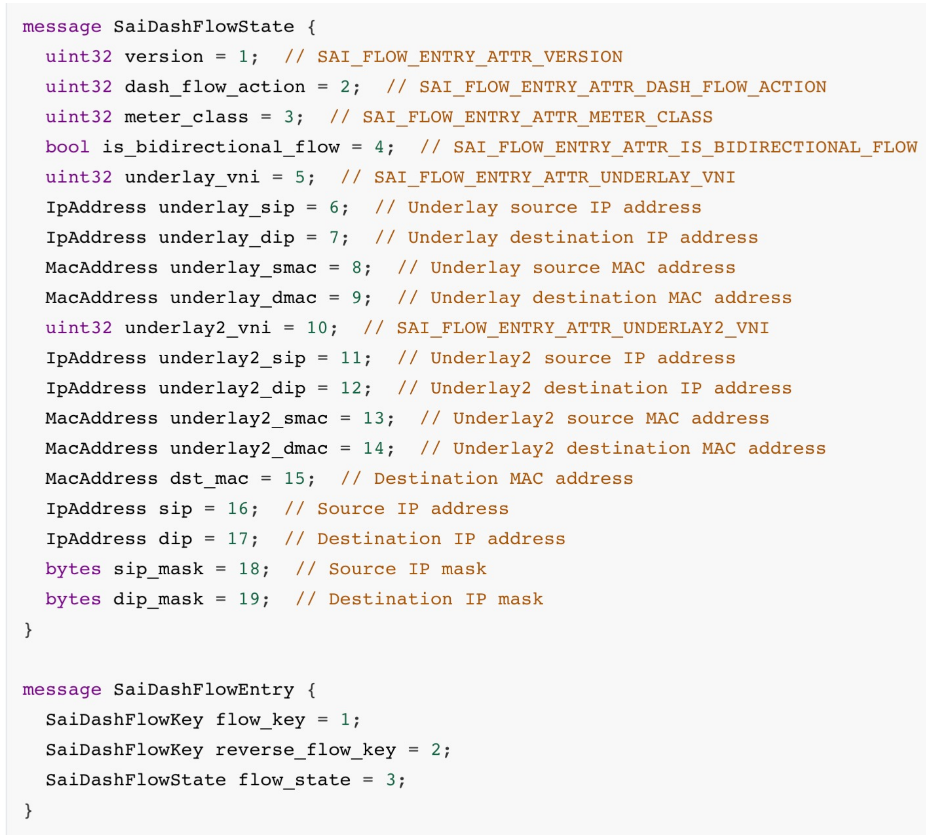
# We iterate filter definition many times. Only sai object can me the sai requirement.

DASH Flow Design – Bulk Get Session Filter

```
typedef enum _sai_dash_flow_entry_bulk_get_session_filter_key_t
{
    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_NONE,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_ENI_MAC,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_IP_PROTO,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_SRC_IP,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_DST_IP,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_SRC_PORT,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_DST_PORT,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_FLOW_VERSION,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_FILTER_KEY_AGED,

} sai_dash_flow_entry_bulk_get_session_filter_key_t;
```

```
typedef enum _sai_dash_flow_entry_bulk_get_session_op_key_t
{
    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_OP_KEY_FILTER_OP_INVALID,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_OP_KEY_FILTER_OP_EQUAL_TO,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_OP_KEY_FILTER_OP_GREATER_THAN,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_OP_KEY_FILTER_OP_GREATER_THAN_OR_EQUAL_TO,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_OP_KEY_FILTER_OP_LESS_THAN,

    SAI_DASH_FLOW_ENTRY_BULK_GET_SESSION_OP_KEY_FILTER_OP_LESS_THAN_OR_EQUAL_TO,

} sai_dash_flow_entry_bulk_get_session_op_key_t;
```

Bulk Sync gRPC Message



```
message SaiDashFlowState {
  uint32 version = 1;  // SAI_FLOW_ENTRY_ATTR_VERSION
  uint32 dash_flow_action = 2;  // SAI_FLOW_ENTRY_ATTR_DASH_FLOW_ACTION
  uint32 meter_class = 3;  // SAI_FLOW_ENTRY_ATTR_METER_CLASS
  bool is_bidirectional_flow = 4;  // SAI_FLOW_ENTRY_ATTR_IS_BIDIRECTIONAL_FLOW
  uint32 underlay_vni = 5;  // SAI_FLOW_ENTRY_ATTR_UNDERLAY_VNI
  IpAddress underlay_sip = 6;  // Underlay source IP address
  IpAddress underlay_dip = 7;  // Underlay destination IP address
  MacAddress underlay_smac = 8;  // Underlay source MAC address
  MacAddress underlay_dmac = 9;  // Underlay destination MAC address
  uint32 underlay2_vni = 10;  // SAI_FLOW_ENTRY_ATTR_UNDERLAY2_VNI
  IpAddress underlay2_sip = 11;  // Underlay2 source IP address
  IpAddress underlay2_dip = 12;  // Underlay2 destination IP address
  MacAddress underlay2_smac = 13;  // Underlay2 source MAC address
  MacAddress underlay2_dmac = 14;  // Underlay2 destination MAC address
  MacAddress dst_mac = 15;  // Destination MAC address
  IpAddress sip = 16;  // Source IP address
  IpAddress dip = 17;  // Destination IP address
  bytes sip_mask = 18;  // Source IP mask
  bytes dip_mask = 19;  // Destination IP mask
}

message SaiDashFlowEntry {
  SaiDashFlowKey flow_key = 1;
  SaiDashFlowKey reverse_flow_key = 2;
  SaiDashFlowState flow_state = 3;
}
```

Fig. Format of gRPC bulk get session message

DASH Flow Design – Bulk Get Session Notification

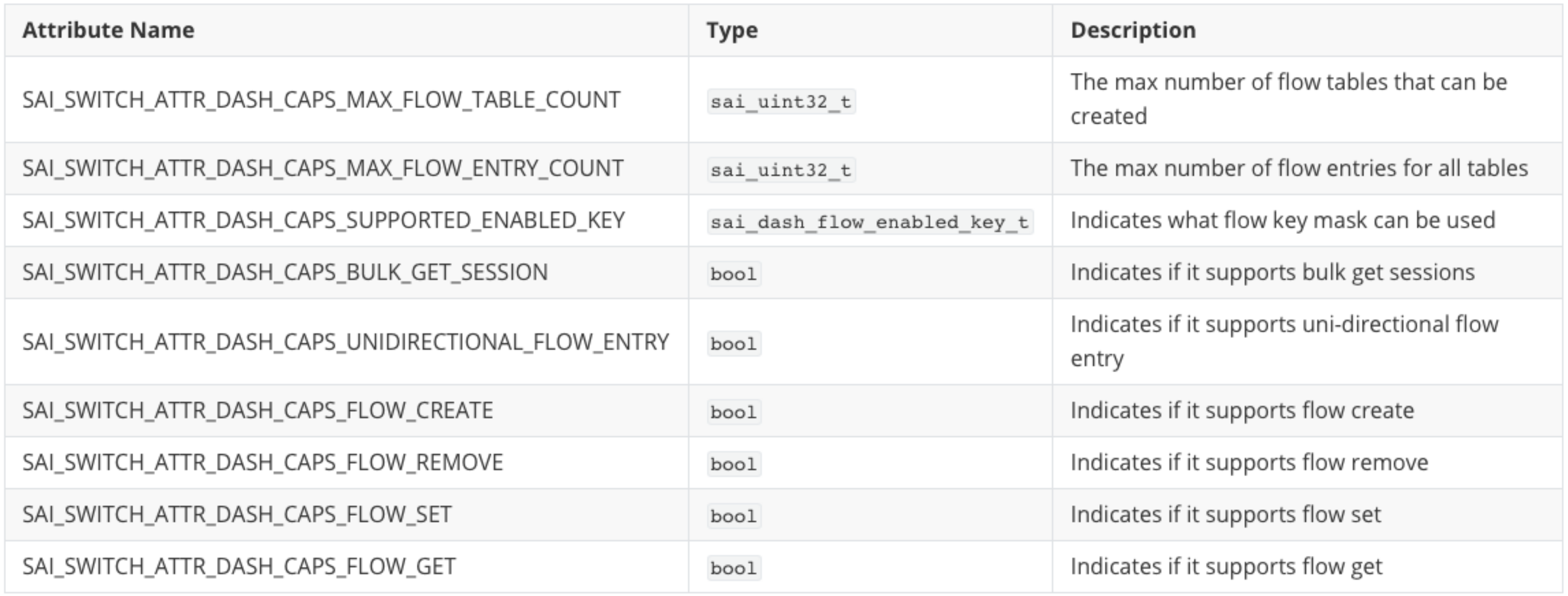
- GRPC target server mode
  - Notify when it is finished
- Flow event notification mode
  - Notify to get new flow
  - Notify when it is finished

```c
/**
 * @brief bulk flow get event type
 */
typedef enum _sai_flow_bulk_get_session_event_t
{
    SAI_FLOW_BULK_GET_SESSION_FINISHED,

    SAI_FLOW_BULK_GET_SESSION_FLOW_ENTRY,

} sai_flow_bulk_get_session_event_t;

/**
 * @brief Notification data format received from SAI HA set callback
 *
 * @count attr[attr_count]
 */
typedef struct _sai_flow_bulk_get_session_event_data_t
{
    sai_flow_bulk_get_session_event_t event_type;

    sai_object_id_t flow_bulk_session_id;

    sai_flow_entry_t *flow_entry;

    uint32_t attr_count;

    sai_attribute_t *attr_list;

} sai_flow_bulk_get_session_event_data_t;
```

Support Protobuf flow programming

- Use ”SAI_FLOW_ENTRY_ATTR_FLOW_DATA_PB” attribute
- Difference
  - Attribute
    - Incremental set/get
      - Flexible
  - Protobuf
    - Complete set/get
      - Efficient

DASH Flow Design – Capability



Attribute Name	Type	Description
SAI_SWITCH_ATTR_DASH_CAPS_MAX_FLOW_TABLE_COUNT	`sai_uint32_t`	The max number of flow tables that can be created
SAI_SWITCH_ATTR_DASH_CAPS_MAX_FLOW_ENTRY_COUNT	`sai_uint32_t`	The max number of flow entries for all tables
SAI_SWITCH_ATTR_DASH_CAPS_SUPPORTED_ENABLED_KEY	`sai_dash_flow_enabled_key_t`	Indicates what flow key mask can be used
SAI_SWITCH_ATTR_DASH_CAPS_BULK_GET_SESSION	`bool`	Indicates if it supports bulk get sessions
SAI_SWITCH_ATTR_DASH_CAPS_UNIDIRECTIONAL_FLOW_ENTRY	`bool`	Indicates if it supports uni-directional flow entry
SAI_SWITCH_ATTR_DASH_CAPS_FLOW_CREATE	`bool`	Indicates if it supports flow create
SAI_SWITCH_ATTR_DASH_CAPS_FLOW_REMOVE	`bool`	Indicates if it supports flow remove
SAI_SWITCH_ATTR_DASH_CAPS_FLOW_SET	`bool`	Indicates if it supports flow set
SAI_SWITCH_ATTR_DASH_CAPS_FLOW_GET	`bool`	Indicates if it supports flow get

Can support partial implantation.

Use case: NAT Gateway

NAT Gateway flow managment

Flow learning

•DPU1 data plane: new flow arrived(upon FT lookup miss), trapped to ARM
•DPU1 ARM: lauch lookup into resources based on direction, then create flow with **FLOW APIs**; apply packet actions and send it out via control plane injection
•DPU1 ARM: Send out flow meta to DPU2 for H/A(add). DPU2 ARM: add flow using **FLOW APIs**

Flow aging

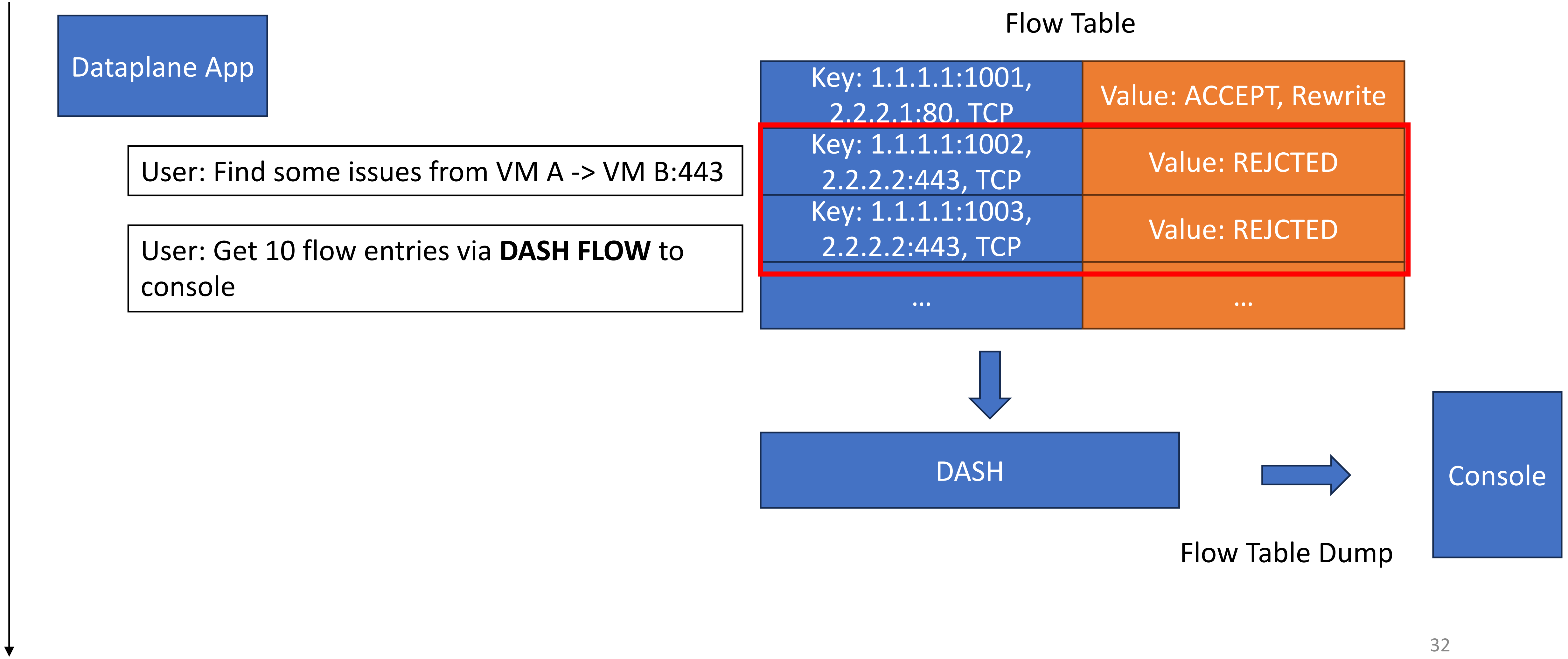
•DPU1 ARM: bulk get flow from data plane using **FLOW APIs** with filters(aged)
•DPU1 ARM: delete flow using **FLOW APIs**, update resources usage
•DPU1 ARM: Send out flow meta to DPU2 for H/A(del), DPU2 ARM: del flow using **FLOW APIs**

Flow sync for recovery

•DPU1 recover, DPU2 ARM bulk get flow from data plane using **FLOW APIs** with filters None(all flow)
•DPU2 ARM receive sync message, add flow using **FLOW APIs**
•DPU2 recover process is similar

Use case: Flow Diagnose

Use Case: Flow Diagnose



User: Find some issues from VM A -> VM B:443

User: Get 10 flow entries via **DASH FLOW** to console

Flow Table

Key: 1.1.1.1:1001, 2.2.2.1:80, TCP	Value: ACCEPT, Rewrite
Key: 1.1.1.1:1002, 2.2.2.2:443, TCP	Value: REJCTED
Key: 1.1.1.1:1003, 2.2.2.2:443, TCP	Value: REJCTED
…	…

Use case: Flow-level HA

Use case: Flow-level HA - New flow

DPU 1: Active

Dataplane: New flow arrived, fwd to ARM

ARM: New flow, Compute the state

ARM: Create new flow entry via **DASH Flow API**

Dateplane: Receive the flow, send it to DPU2 via dataplane

DPU 2: Stand by

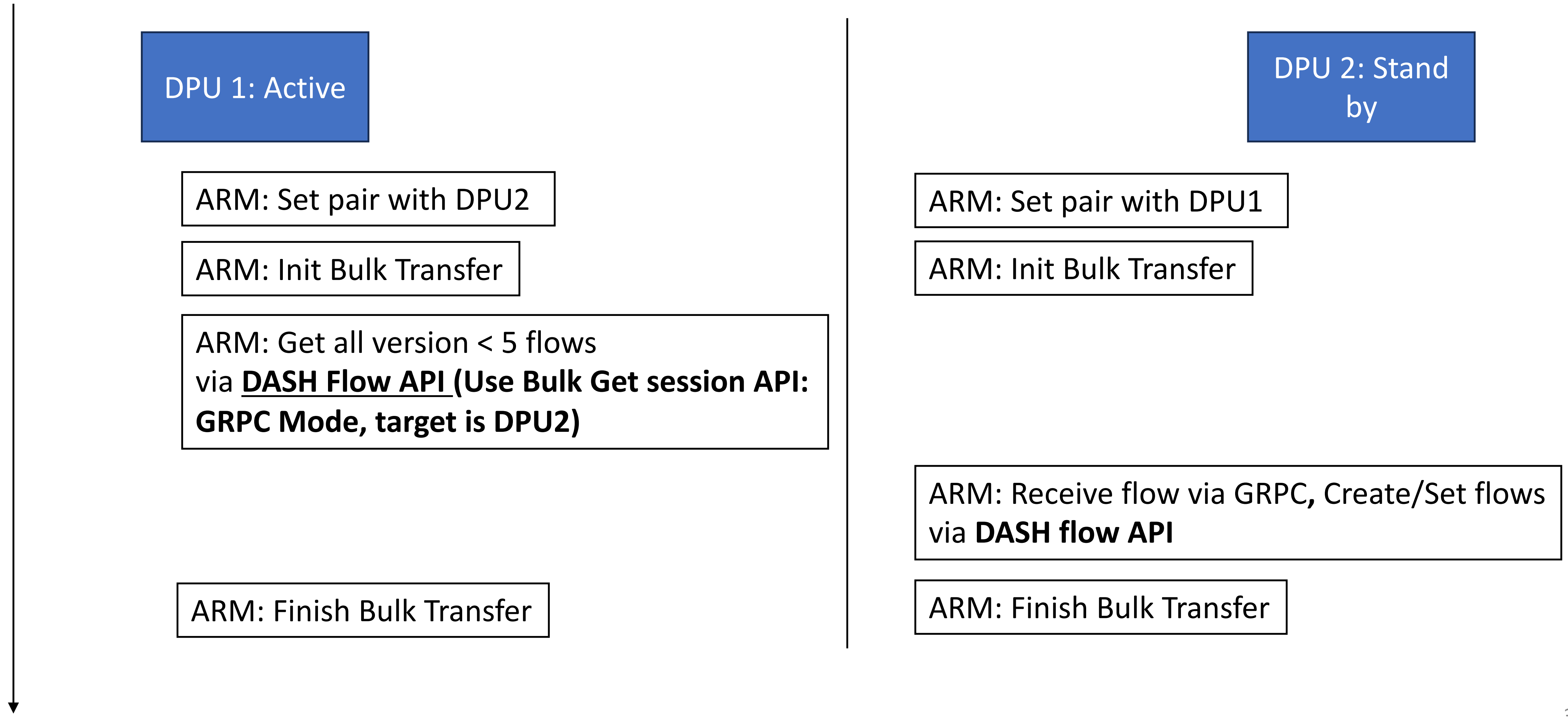
Dataplane: New flow arrived, fwd to ARM

ARM: New flow, Compute the state

ARM: Create new flow entry via **DASH Flow API**

Dateplane: Receive the flow, send it back to DPU1

Use case: Flow-level HA – Bulk Transfer



DPU 1: Active

- ARM: Set pair with DPU2
- ARM: Init Bulk Transfer
- ARM: Get all version < 5 flows via **<u>DASH Flow API</u> (Use Bulk Get session API: GRPC Mode, target is DPU2)**
- ARM: Finish Bulk Transfer

DPU 2: Stand by

- ARM: Set pair with DPU1
- ARM: Init Bulk Transfer
- ARM: Receive flow via GRPC**,** Create/Set flows via **DASH flow API**
- ARM: Finish Bulk Transfer

Summary

- Standard DASH flow to support millions of flow states.
- Vendor-neutral flow management
- Support different scenarios
- Flexible design
    - Uni/bi-direction flow
    - Different key masks
    - All SDN packet transformation currently defined in DASH
    - Bulk get session (event, grpc) with up to 5 filters
    - Protobuf flow programming