MTU Settings in SONiC

Introduction 2

Overview 2

Requirements 2

Assumptions 2

Port MTU 3

Host Interface MTU 3

APP DB Schema 3

SAI Attribute 5

SAI API 5

IP MTU 6

APP DB Schema 6

SAI Attribute 6

SAI API 7

Limitations 7

MTU Settings Design 7

References 8

# Introduction

## Overview

This document describes maximum transmission unit (MTU) configurations and behaviors in SONiC. In computer networking, the MTU of a communication protocol of a layer defines the size, in bytes, of the largest protocol data unit that the layer is allowed to transmit over one interface. One MTU is associated with each interface, layer and protocol.

In SONiC, there are two type of MTU for an interface based on layers.

1. Port MTU (Layer 2): the largest number of bytes that can be carried by an Ethernet frame.
2. IP MTU (Layer 3): the maximum size of an IP payload allowed to be transmitted over an L3 interface, excluding the L2 header and trailer.

## Requirements

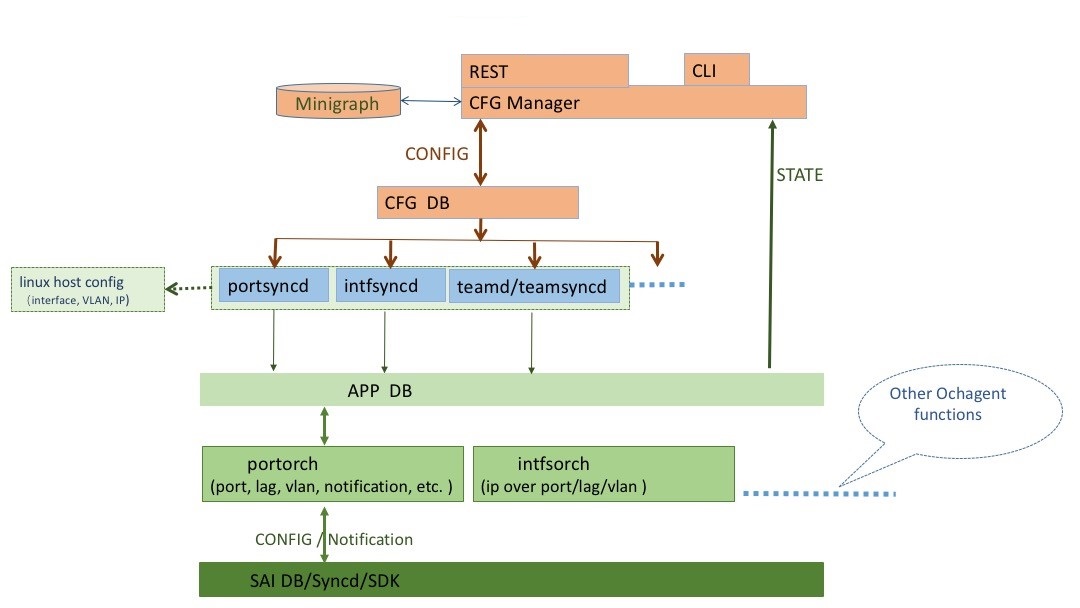
1. The defaulted MTU is 1500, with a configurable range from 68 bytes to 9216 bytes. The minimum Ethernet Frame is 64 bytes, but our linux host can only take 68 bytes as minimum MTU. So we choose 68 bytes as the lower bound.
2. The MTU configured on ASIC is 22 Bytes more than the MTU configured on corresponding host interface, which includes the overhead of L2 header (14 Bytes), FCS header (4 Bytes) and Vlan tag (4 Bytes).
3. Port MTU and IP MTU can be configured with different values on the same interface. However, IP MTU should be capped by the Port MTU. Otherwise the IP packet larger the port MTU may still go out of the host interface, but get dropped by the ASIC.

## Assumptions

1. The Port MTU and IP MTU settings are configured through the new Config DB model.

# Port MTU

Port MTU is configured by either CLI or Restful API talking to CFG manager, who updates the corresponding port objects inside CFG DB. Then portsyncd will set the MTU of the host interface via ip link set/ifconfig command under Linux environment. Upon success, portsyncd will push the config into PORT\_TABLE inside APP DB. If there is any change with PORT\_TABLE, its subscriber, PortsOrch, will add an extra 22 Bytes to the corresponding MTU, write into SAI DB and program the ASIC.



## Host Interface MTU

admin@sonic:~$ sudo ifconfig Ethernet0

Ethernet0 Link encap:Ethernet HWaddr 00:05:64:30:17:ec

inet addr:10.1.0.1 Bcast:10.1.0.255 Mask:255.255.255.0

UP BROADCAST MULTICAST MTU:1500 Metric:1

RX packets:0 errors:0 dropped:0 overruns:0 frame:0

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

## APP DB Schema

### PORT\_TABLE

Stores information for physical switch ports managed by the switch chip.

;Defines layer 2 ports

;In SONiC, Data is loaded from configuration file by portsyncd

;Status: Mandatory

port\_table\_key = PORT\_TABLE:ifname ; ifname must be unique across PORT,INTF,VLAN,LAG TABLES

device\_name = 1\*64VCHAR ; must be unique across PORT,INTF,VLAN,LAG TABLES and must map to PORT\_TABLE.name

admin\_status = BIT ; is the port enabled (1) or disabled (0)

oper\_status = BIT ; physical status up (1) or down (0) of the link attached to this port

lanes = list of lanes ; (need format spec???)

ifname = 1\*64VCHAR ; name of the port, must be unique

mtu = 1\*4DIGIT ; MTU for the interface

mac = 12HEXDIG ;

example:

127.0.0.1:6379> hgetall PORT\_TABLE:Ethernet0

1) "lanes"

2) "1"

3) "alias"

4) "Ethernet0"

5) "oper\_status"

6) "down"

7) "admin\_status"

8) "down"

9) "mtu"

10) "1500"

Please note, the same schema is applicable for LAG. So the Port MTU can also be configured on LAG, and each LAG member ports will inherit the port MTU from LAG.

For example, we configured a LAG team0 with 3 member ports Etherent10, Ethernet11 and Ethernet12 with default MTU 1500. After changing the MTU of team0, the member port MTU will also get updated correspondingly.

admin@sonic:~$ sudo ifconfig team0 mtu 1600

admin@sonic:~$ redis-cli -n 0 keys \*LAG\*

1) "LAG\_MEMBER\_TABLE:team0:Ethernet10"

2) "LAG\_MEMBER\_TABLE:team0:Ethernet12"

3) "LAG\_MEMBER\_TABLE:team0:Ethernet11"

4) "LAG\_TABLE:team0"

admin@sonic:~$ redis-cli -n 0 hgetall LAG\_TABLE:team0

1) "admin\_status"

2) "down"

3) "oper\_status"

4) "down"

5) "mtu"

6) "1600"

admin@sonic:~$ redis-cli -n 0 hgetall "PORT\_TABLE:Ethernet10"

1) "lanes"

2) "11"

3) "alias"

4) "Ethernet10"

5) "oper\_status"

6) "up"

7) "admin\_status"

8) "up"

9) "mtu"

10) "1600"

admin@sonic:~$ redis-cli -n 0 hgetall "PORT\_TABLE:Ethernet11"

1) "lanes"

2) "12"

3) "alias"

4) "Ethernet11"

5) "oper\_status"

6) "up"

7) "admin\_status"

8) "up"

9) "mtu"

10) "1600"

admin@sonic:~$ redis-cli -n 0 hgetall "PORT\_TABLE:Ethernet12"

1) "lanes"

2) "21"

3) "alias"

4) "Ethernet12"

5) "oper\_status"

6) "up"

7) "admin\_status"

8) "up"

9) "mtu"

10) "1600"

## SAI Attribute

sai\_attribute\_t attr;

attr.id = SAI\_PORT\_ATTR\_MTU;

attr.value.u32 = mtu;

## SAI API

sai\_status\_t status = sai\_port\_api->set\_port\_attribute(id, &attr);

# IP MTU

Similarly, the IP MTU is also configured through CLI/Restful API which sets if\_mtu value of the router interface objects inside CFG DB. Then Intfsyncd copies the corresponding change into APP DB. Upon receiving update notification from APP DB, IntfsOrch pushes the if\_mtu value into SAI DB, and adds the MTU attribute to the attribute list during the sai router interface creation.

## APP DB Schema

### INTF\_TABLE

intfsyncd manages this table. In SONiC, CPU (management) and logical ports (vlan, loopback, LAG) are declared in /etc/network/interface and loaded into the INTF\_TABLE.

IP prefixes are formatted according to [RFC5954](https://tools.ietf.org/html/rfc5954) with a prefix length appended to the end

;defines logical network interfaces, an attachment to a PORT and list of 0 or more

;ip prefixes

;

;Status: stable

key = INTF\_TABLE:ifname:IPprefix ; an instance of this key will be repeated for each prefix

IPprefix = IPv4prefix / IPv6prefix ; an instance of this key/value pair will be repeated for each prefix

scope = "global" / "local" ; local is an interface visible on this localhost only

if\_mtu = 1\*4DIGIT ; MTU for the interface

family = "IPv4" / "IPv6" ; address family

example:

127.0.0.1:6379> hgetall INTF\_TABLE:Ethernet0:10.1.0.1/24

1) "scope"

2) "global"

3) "family"

4) "IPv4"

5) "if\_mtu"

6) "1500"

## SAI Attribute

if\_attr.id = SAI\_ROUTER\_INTERFACE\_ATTR\_MTU;

rif\_attr.value.u32 = mtu;

rif\_attrs.push\_back(rif\_attr);

## SAI API

sai\_status\_t status = sai\_rif\_api->create\_router\_interface(&rif\_id, rif\_attrs.size(), rif\_attrs.data());

## Limitations

1. SAI\_ROUTER\_INTERFACE\_ATTR\_MTU is not handled in current brcm\_sai. So the asic is not programmed correctly with the ip mtu. Have provided the fix to Broadcom.
2. There is no attribute update path for router interface in current SONiC. So the ip mtu can only being set during the router interface creation. Needs to implement set\_router\_interface\_attribute and get\_router\_interface\_attribute after config DB and interface schema are in place.

# MTU Settings Design

If the Kernel MTU is configured on the linux host interface, the corresponding port will use the Kernel MTU with an additional 22 Bytes as SAI\_PORT\_ATTR\_MTU. Any router interface created on the port, will use Kernel MTU as SAI\_ROUTER\_INTERFACE\_ATTR\_MTU.

Summary:

Kernel MTU = IP MTU = Port MTU – 22 Bytes

**Config DB Schema**

"PORT": {

"Ethernet8": {

"alias": "Ethernet9",

"mtu": "1500"

},

…

},

"PORTCHANNEL": {

"PortChannel10": {

"fallback": "true",

"members": [

"Ethernet10"

],

"mtu": "1500"

},

…

},

# References

1. SONiC Configuration Management
2. <https://github.com/Azure/SONiC/wiki/Architecture>
3. Add config check in config manager to reject invalid config
4. Apply ip mtu on both linux host interface and SAI\_ROUTER\_INTERFACE
5. Trap frames exceeding MTU to CPU?