BCM56070

Switch Programming Guide

Programming Guide

Table of Contents

Chapter 1: Introduction

This document describes the features and architecture of the Broadcom® BCM56070 family of highly integrated Ethernet switches. The BCM56070 family includes multiple devices, each with a different part number. For information about the I/O bandwidth, throughput, and port configurations that each device in the BCM56070 family supports, refer to the BCM56070 data sheet (56070-DS1xx).

All features described in this document refer to the BCM56070 base device. For the latest package information and device features, refer to the device data sheet. If any discrepancies exist between this document and the data sheet, the data sheet takes precedence.

Chapter 2: Architectural Overview

The BCM56070 device is built using a modular, high-performance, pipeline packet-switching architecture.

This unique structure has several benefits, which include the following:

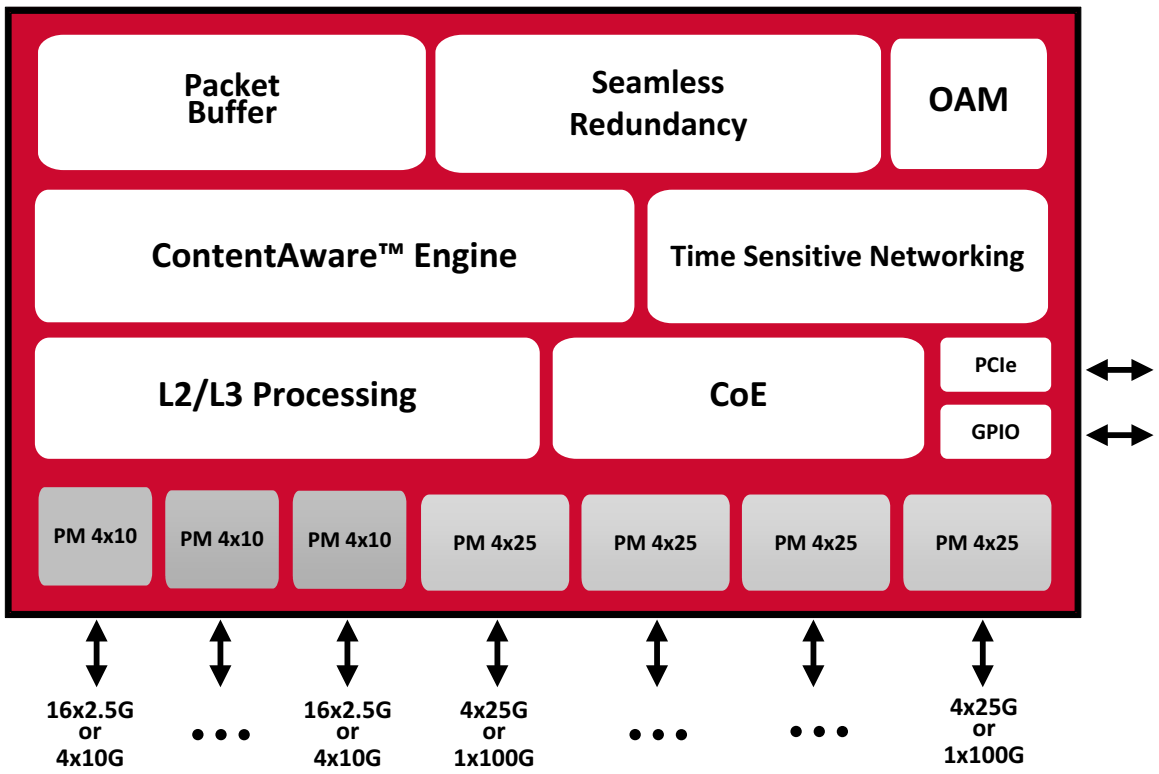
- Flexible port configurations.
- Scalable throughput.
- Scalable custom features.
- Migration to different process technologies without architectural changes.

2.1 Device Functional Overview

The following figure illustrates the BCM56070 device functions.

The device can support up to 64 front-panel ports, all running at line rate.

Figure 1: Device Block Diagram



In general, the device is equipped with packet parsing capabilities, L2 and L3 lookup and forwarding functionalities (including channelized flow handling), and time sensitive networking features, such as seamless redundancy, time aware filter, policing, and scheduling. Other key features are the search engines that constitute of hashing algorithm, binary search, and CAM search. With the device's ContentAware™ engine, complex ACL processing is easily achieved.

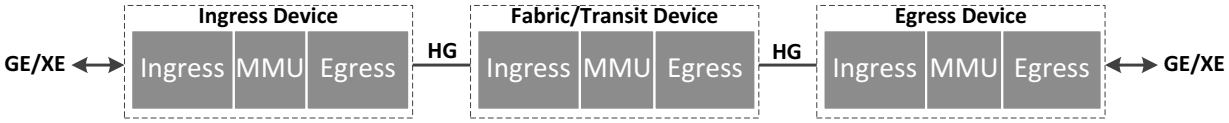
The Memory Management Unit (MMU) handles all queuing and scheduling functions. The MMU is where time aware scheduling is implemented. The device has built-in 2.0 MB of memory for packet switching.

All lookups, such as L2, L3, and ACL tables, are done at the ingress pipeline, while all packet modifications are done in the egress pipeline.

2.2 System View

The following figure shows the BCM56070 system view.

Figure 2: System Diagram



The Ingress Device performs the following functions:

- Parse the first 128B of the packet.
- L2 and L3 lookup using the packet headers.
- Flexible ACL processing using eight ACL database implemented in TCAM.
- Metering and statistics on per ACL basis.
- Packet buffering, admission control, and scheduling.
- Packet modifications can be applied for certain packet types.

The Fabric or Transit Device performs the following functions:

- Switch the packet based on HiGig2™ header information.
- Perform spatial multicast.

The Egress Device performs the following functions:

- Parse the HiGig header or the first 128B of the packet.
- Lookup using the HiGig™ header keys to determine the local egress port.
- Packet buffering, admission control, and scheduling.
- Packet modifications.
- Flexible ACL processing using internal TCAM.
- Metering and statistics on a per ACL basis.

Chapter 3: Device Interfaces

3.1 Port Configurations

The BCM56070 supports flexible port configurations, including the following typical configurations:

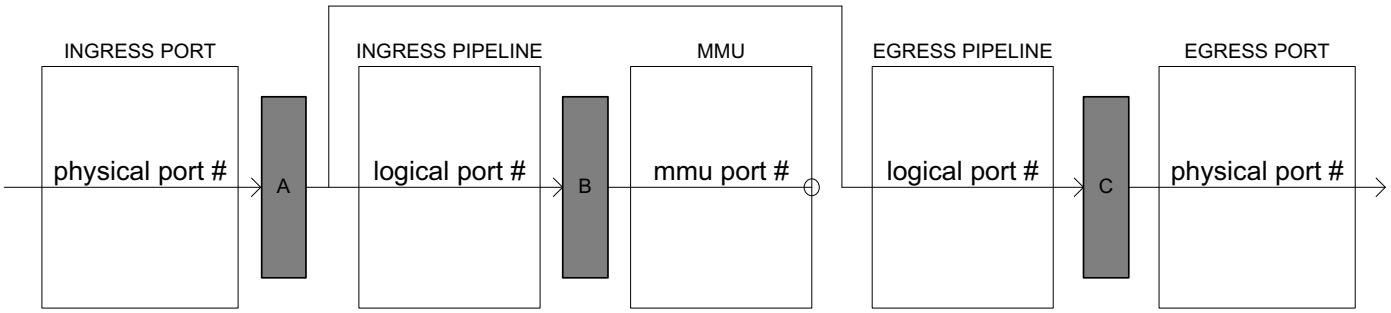
- (48 × 2.5GbE) + (6 × 50GbE)
- (12 × 10GbE) + (6 × 50GbE)

The device can support a mixture of 1G, 2.5G, 10G, 25G, 40G, 50G, and 100G ports with a maximum of 48 front-panel ports.

3.2 Port Numbering

As the following figure shows, the BCM56070 port numbering view depends on the functional block.

Figure 3: Port Numbering within the Device



3.2.1 Physical Port Number

The physical ports are the interfaces exposed to the user and are essentially pins from the SerDes cores. The BCM56070 includes two types of SerDes cores: Merlin (TSC4-MC) and Falcon (TSC4-FC). Each type of SerDes core is capable of supporting a range of port speeds.

3.2.2 Physical Port to Logical Port

The logical port number is the port name used in the ingress pipeline (IP) and egress pipeline (EP). The logical port number range is from 0 to 127.

On the ingress, each physical port is mapped to a logical port that is used by the IP and EP.

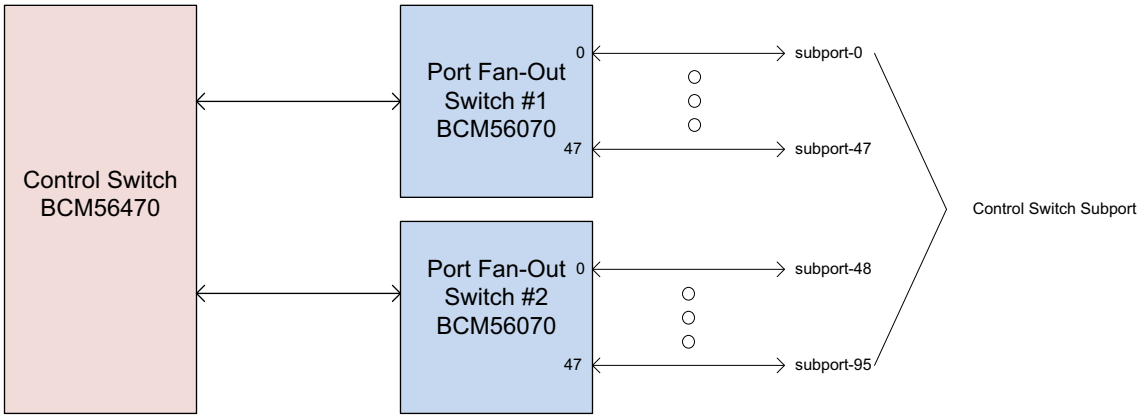
3.2.3 EP Logical Port to Physical Port Number

The functional blocks within the EP operate on the logical port number. These logical port numbers are mapped back to a physical port for transmission.

3.2.4 Port Numbering for Channelized Applications

For channelized applications, each subport is represented as a system port on the control switch for packet forwarding. Each BCM56070 operating in channelized port-fan-out mode can support 48 subports, also known as channels. The system-wide subport number is not known to the BCM56070; instead, it is a property of the control switch.

Figure 4: Channelized Application Diagram



3.3 PCIe

The PCI Express (PCIe) interface provides the connectivity to the local host CPU for local configuration, management, and control of the device, as well as for debugging purposes. For details on the PCIe protocol, refer to the PCIe specifications published by PCI-SIG (http://www.pcisig.com).

For electrical characteristics, strapping pin configurations, and operation modes, refer to the BCM56070 data sheet.

The PCIe interface supports only the endpoint role. It is compliant to Gen1, Gen2, and Gen3 of the PCIe interface standard and supports a configurable speed per lane (2.5G, 5G, or 8G). In PCIe Gen3 mode, microcode from an external QSPI must be loaded during boot-up of the device.

3.4 LED Interface

For information about the LED interface, refer to the BCM56070 data sheet.

3.5 Clock Sources

The BCM56070 has several clock inputs, which are shown in the following table. The table also lists the internal block that the corresponding PLL drives.

Table 1: BCM56070 Clock Inputs

Clock Source	Input Frequency	Comments
XTAL P/N	50 MHz	Provides clocks to IPROC, IP, EP, MMU
TS_PLL_REFCLK	50 MHz	Alternative clock sources when needed.
BS0_PLL_REFCLK	50 MHz	Alternative clock sources when needed.
BS1_PLL_REFCLK	50 MHz	Alternative clock sources when needed.
LCPLL0	156.25 MHz	PM4x10Q and PM4x25.
PCIe_REFCLK	100 MHz	PCIe

3.6 MDC and MDIO Information

The CPU Management Interface Controller (CMIC) supports the IEEE 802.3u standard MIIM interface, which is a two-wire serial bus controlled by the CMIC that allows register access to all the PHYs in the system. This interface is used to read or write data from the PHY. The two signals for MIIM are MDC (clock) and MDIO (bidirectional data). The CPU programs the internal and external PHY registers using this interface. The MIIM interface can be configured to support Clause 22 or Clause 45 and can operate at 2.5 MHz. For the maximum frequency supported by this interface, refer to the BCM56070 data sheet.

By default, the device is configured as the MDIO master. Two MDIO interfaces can be used to communicate with an external PHY. Each interface has a unique bus ID and can support a fixed range of PHY addresses.

Table 2: MDC/MDIO Signal Information

Signal	Frequency	Description
MDC_0 MDIO_0	2.5MHz	MIIM Clock and Data Interface 0 External PHY address: 0x00–0x1F
MDC_1 MDIO_1	2.5MHz	MIIM Clock and Data Interface 1 External PHY address: 0x20–0x3F

3.7 Network Time Synchronization

Network time synchronization is a method to distribute time synchronously across the network. One grand master clock provides the reference time to directly attached slave devices. The slave devices then continue propagating the time down the hierarchy to other devices on the network.

Multiple protocols exist for distributing the time across networks. Both IEEE 802.1AS and IEEE 1588v2 are supported for time distribution. The chip requirements to support these two protocols are largely similar with the exception of the packet formats.

After the local time is synchronized, it can be used to either drive a synchronized time value off-chip using the BroadSync® interface or to control internal time-based scheduling mechanism. Ethernet AV scheduling is also supported.

3.7.1 BroadSync Timing Interface

This interface provides a means for a switch device to provide its clock information to external functions or to be supplied with clock information by external functions.

The devices that make up a time-aware bridged network operate in one of three modes:

- Timing master
- Peer-to-peer transparent clock (TC)
- Timing slave

Timing masters are used by all of the timing slave devices as the master time reference, and their clocks are synchronized to that of the master's.

The peer-to-peer transparent clocks are not consumers of timing information, but are critical in distributing timing information from the master to the slaves. It is possible, however, for a peer-to-peer transparent clock to also be a timing slave, that is, a consumer of timing information.

In some applications, it is required for the timing master to receive its clock information from an external source such as a GPS receiver or an atomic clock. Consequently, the BroadSync interface is an input for a timing master; obtaining timing information from an external source.

Similarly, timing slaves must be able to convey clock information to hardware that is implemented alongside the timing slave devices. Consequently, the BroadSync interface is an output; conveying timing information to an external consumer of timing information.

3.7.2 Using the BroadSync Interface

The BroadSync interface is used by ordinary clocks (OCs) in either a master or slave role. When the OC is a master, the BroadSync interface is configured as an input; accepting timing information from external hardware. When the OC is a slave, BroadSync is configured as an output; providing timing information to external hardware.

3.7.3 Master Mode: Timing Input

External hardware provides the bitClock and heartBeat signals. During each heartBeat period, the external hardware also shifts in the timeCode values, consisting of the 80-bit time value and 8-bit accuracy value. The time value shifted in corresponds to the time of the most recent rising edge of the heartBeat signal. The internal time value is calibrated to the external signals through the following process:

1. The rising edge of heartBeat is used to sample the device's internal free-running clock value.
2. The sampled free-running clock value is compared to the time value subsequently shifted in using the timeCode signal.
3. These pairs of values (shifted-in time and sampled free running time) are provided to the CPU on an occasional basis.
4. The differences and rates of change of the differences of the two time bases are used to derive drift and offset compensation values.
5. The computed drift and offset compensation values are used to correct the free running clock-based time stamp values for use in the follow-up messages. These correction factors are also used by the hardware-based clock compensation and generation function to synthesize various frequencies for use on chip.

For very high-precision applications, the bitClock input from the external time source may be multiplied up to a workable frequency by a precision PLL and used to drive the free-running clock: doing so eliminates the need for drift compensation and requires only offset compensation to derive very precise time information.

3.7.3.1 Slave Mode: Timing Output

External hardware accepts the bitClock and heartBeat signals and uses them to synchronize its behavior with that of the grand master. The bitClock signal may be used directly, it may be filtered and stabilized by an external PLL or it may be multiplied to some more suitable frequency.

The values shifted out using the timeCode signal may be used by the external hardware if time of day information is required. To accomplish this, the external hardware must sample its own local time of day value at the rising edge of heartBeat. The difference between that sampled local time and the time value provided using BroadSync is then used as an offset to correct the local time to match the time at the grand master.

3.7.4 Device-Specific Support

Refer to the following sections for information on device-specific support.

3.7.4.1 Feature List

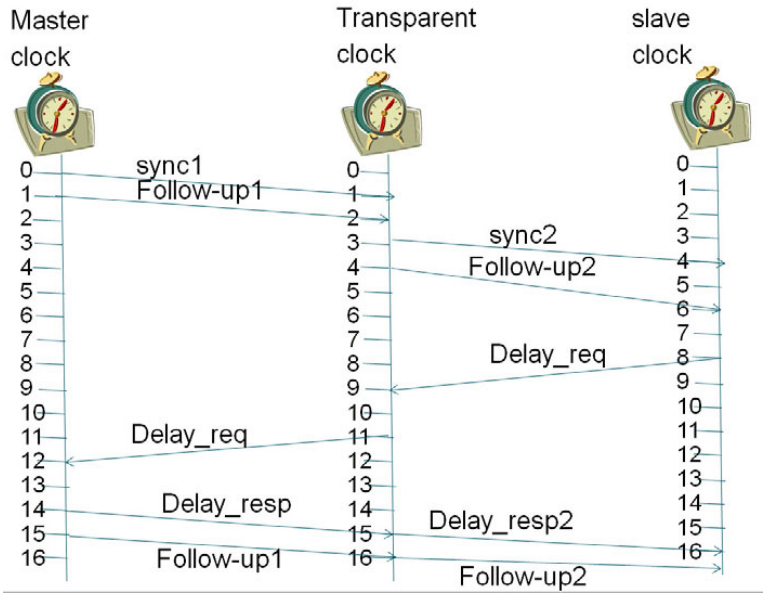
The following components of network time synchronization are supported:

- IEEE 802.1AS and IEEE 1588v2 Network Time Distribution (packet time stamping).
- Ethernet AV scheduling.
- BroadSync Interface..

3.7.4.2 1-Step

1-Step TC is a TC that adds the residence time to the correctionField of the event message by the egress port while fixing the Cyclic Redundancy Check (CRC checksum) of the Ethernet packet.

Figure 5: Synchronization Principle, 1-Step TC



No additional software is required to support 1-Step TC.

3.7.4.3 2-Step

This section describes the 2-Step TC.

3.7.4.3.1 Receive Processing

All RX packets, including all time synchronization control packets, will be timestamped by the module's local timer. The local timer will be sampled when the SOP of the packet arrives on the MAC interface (MII, GMII, XGMII).

The module's local timer will be a free-running counter based off a 250-MHz reference clock. All of the local timers across the chip are reset synchronously. This means that when a hardware reset occurs, all of the local timers need to come out of reset synchronously. It is also expected that the free running local timer will not be changed via a block-level software reset. This is to ensure that all of the free running counters spread across the different port modules are running synchronously (off the same clock and incrementing with the same values).

3.7.4.3.2 Transmit

If the TIMESTAMP indicator is set when the first byte of the outgoing packet is transmitted by the MAC, then the port module's local timestamp value is stored in a per-port FIFO containing four entries. After the first packet is transmitted out of the port's MAC, the software will read the timestamp from the FIFO for the first packet and insert this timestamp value for the follow up messages.

Each MAC will contain a TX_TIMESTAMP_FIFO that is accessed using the TX_TS_DATA register. The FIFO is four entries deep and stores 32-bit timestamp values. Timestamps are only loaded into this FIFO for packets that have the TIMESTAMP indicator set. If the TX_TIMESTAMP_FIFO becomes full and a new packet is received with the TIMESTAMP indicator set, the TX_TIMESTAMP_FIFO is not updated. The availability of FIFO condition is located in TS_STATUS_CNTRL register. This register shows whether the FIFO is full or empty.

After the software sends a time sync packet, the software will periodically poll the TS_STATUS_CNTRL register.

3.7.5 Packet Formats

The following time synchronization packet formats are supported:

- IEEE 802.1AS
- IEEE 1588v2

3.7.5.1 Detecting IEEE 802.1AS Packets

IEEE 802.1AS sync and delay response packets can be detected by observing a packet's EtherType[15:0] value. If an EtherType[15:0] value of 16'h88f7 is found, then the packet is a network time sync packet. Differentiating sync and delay response messages can be accomplished by examining the msgType[3:0] value.

3.7.5.2 Detecting IEEE 1588v2 Packets

IEEE 1588v2 may operate either at layer 2 or at layer 4. The sync and delay packets can be detected by observing a packet's EtherType[15:0] or UDP destPort[15:0] value. For layer 2, MAC destination address value 0x01-1B-19-00-00-00 is used for all messages except for delay measurement messages, while 0x01-80-C2-00-00-0E is used for delay measurement messages. For all IEEE 1588v2 messages conveyed as layer 2 payloads, an EtherType value of 0x88F7 is used.

When IEEE 1588v2 is operating as a UDP payload, a destPort[15:0] value of 319 is used for either sync or delay response packets, while the UDP destPort[15:0] value of 320 is used for all other IEEE 1588v2-related time synchronization messages. IP multicast addresses are used for all layer 4 IEEE 1588v2 PTP packets. The default IP multicast destination address is 224.0.1.129. Three alternate addresses are 224.0.1.130, 224.0.1.131 and 224.0.1.132.

3.7.6 Pdelay

The exchange of peer delay (Pdelay) request and response messages between multiple systems to synchronize clocks throughout the local area network is governed by the Precision Time Protocol (PTP). This feature is part of the IEEE 1588 specification.

This section describes the ability of the BCM56070 to respond to IEEE Peer-to-Peer (P2P) Pdelay request (Pdelay_Req) messages.

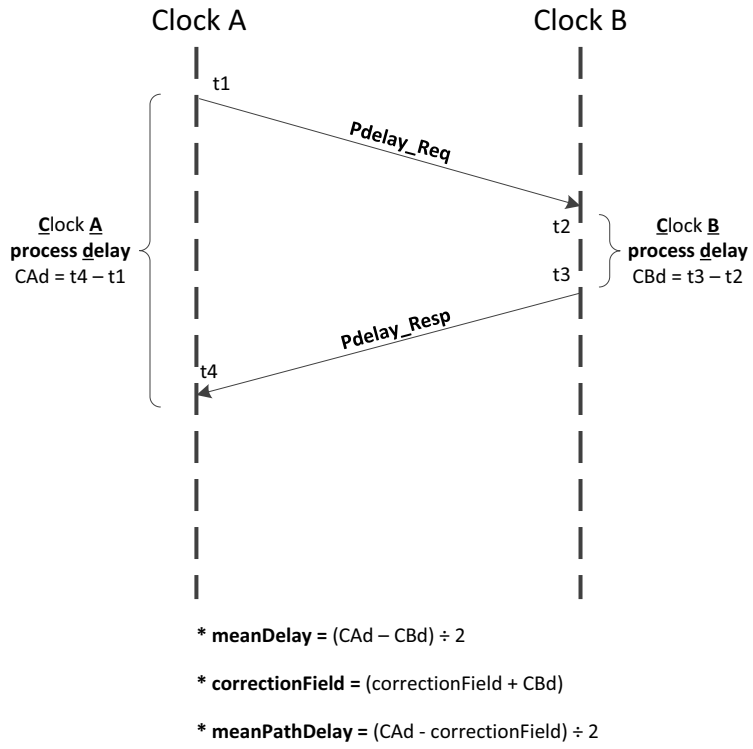
The implementation of the Pdelay response message by the BCM56070 is consistent with the transparent clocking specification that was added in the 2008 version of IEEE 1588.

3.7.6.1 Pdelay Timestamping

As diagrammed in the following steps are taken for sending and receiving a Pdelay_Req message and the corresponding Pdelay_Resp message.

- Clock A stores timestamp t1 when sending Pdelay_Req.
- Clock B records timestamp t2 when receiving Pdelay_Req.
- Clock B records timestamp t3 when sending Pdelay_Resp.
  - requestReceiptTimestamp = t2.
  - correctionField = correctionField + CBd.
  - (For a one-step clock) Clock B sends a new correctionField in Pdelay_Resp.
- Clock A records timestamp t4 when receiving Pdelay_Resp.
- Clock A calculates meanPathDelay.
  - meanPathDelay = (CAd – correctionField) ÷ 2

Figure 6: Pdelay_Req and Pdelay_Resp Message Flow



In a functioning system, both switches at either end of a P2P link act as both Clock A and as Clock B, and both switches send Pdelay_Req frames and respond to Pdelay_Req frames.

In this device, the Clock B role of responding to a Pdelay_Req frame with a corresponding Pdelay_Resp frame is implemented in hardware. However, the generation of Pdelay_Req frames as well as the reception of Pdelay_Resp frames (by the Clock A role) are handled in software.

3.7.6.2 Packet Flow – Responding to Pdelay_Req in the Clock B Role

Theoretically, when a frame enters the device, a timestamp, t2, is generated and stored by the MAC block. Then the packet is qualified to be a valid IEEE 1588 Pdelay_Req frame in the next block. If the packet's EtherType is 0x88F7, the parser will extract the PTP header fields, such as the PTP version and the message type. Once the packet is determined to be a Pdelay_Req message, a PDELAY_REQ bit is set to signal to the rest of the pipeline that this protocol packet is to be transformed into a Pdelay_Resp frame, and it will be sent out of the same port through which it came. The message transformation is done at the end of the egress pipeline where the frame's PTP message type is updated to the appropriate value, SA and DA are determined, and a signal is set to tell the MAC to update the correctionField. The MAC will finally generate another timestamp, t3, and compute a new correctionField value and send out a Pdelay_Resp frame.

3.8 Energy Efficient Ethernet

The BCM56070 device supports Energy Efficient Ethernet (EEE) to reduce power consumption by informing the integrated or external PHY to enter a Low Power Idle (LPI) state, during extended idle periods that may exist between packets.

In general, EEE operates in an asymmetric mode. Meaning, the transmit direction and receive direction may enter and exit the LPI state independently. For 1000BASE-T, however, symmetric operation is required in order to truly benefit from EEE.

In symmetric mode, both the transmit and receive paths must be transmitting or receiving sleep symbols before either side will enter the quiet state.

Chapter 4: Forwarding

4.1 Virtual Local Area Network

Virtual local area networks (VLANs) are used to segment different broadcast domains. One typical application of VLAN is to assign one workgroup on one VLAN and another workgroup on a different VLAN. One of the many reasons for this is security or network efficiency. The BCM56070 supports up to 4094 VLANs. It supports IEEE 802.1Q tagged, MAC-based, IP subnet-based, protocol-based, port-based, and flow-based VLANs. Each VLAN supports a group of member ports that are defined in the VLAN_Table and each port can have more than one VLAN associated with it.

The BCM56070 device supports four global outer TPIDs. Each ingress front-panel port, specifies which of the four global outer TPIDs should be used for parsing incoming packets. Multiple outer TPIDs can be specified per ingress port. The tagging status of a packet from a front-panel port is determined as below:

- Double Tagged (DT): A packet is considered double tagged when the following conditions are true:
  - Bytes 12 and 13 of the packet match any one of the ingress port's configured outer TPIDs.
  - Bytes 16 and 17 of the packet match the per-chip configured inner TPID.
- Single Outer-tagged (SOT): A packet is considered single outer-tagged when the following conditions are true:
  - Bytes 12 and 13 of the packet match any one of the ingress port's configured outer TPIDs.
  - Bytes 16 and 17 of the packet do not match the per-chip configured inner TPID.
- Single Inner-tagged (SIT): A packet is considered single inner-tagged when bytes 12 and 13 of the packet do not match any one of the ingress port's configured outer TPIDs but match the per-chip configured inner TPID.
- Untagged: A packet is considered untagged if bytes 12 and 13 of the packet do not match any one of the ingress port's configured outer TPIDs and do not match the per-chip configured inner TPID.

An incoming HiGig packet is always considered outer-tagged. If bytes 12 and 13 of the ingress packet matches the per-chip configured inner TPID, then the packet is considered double tagged, else it is considered single outer-tagged.

Ingress filtering is one of the many security features that the BCM56070 supports. By enabling the EN_FILTER bit on a per port configuration, the device drops the ingress packet if it is not part of the assigned VLAN. For example, if VLAN number 5 has only port 2 and 4 assigned to it, and a packet ingresses on port 3 with a VLAN ID of 5, while the EN_IFILTER bit is set for port 3, the packet will be dropped.

4.1.1 Virtual Route Forwarding

Virtual Route Forwarding (VRF) is also known as VPN routing or VPN forwarding. Virtual Private Networks (VPNs) provide a secure way for customers to share bandwidth over a common backbone network. Each VPN requires its own routing table called the VRF table. Supporting multiple VRF tables allows a switch to support multiple VPNs, where IP addresses can be overlapped among the VPNs. VRF forms virtual packet forwarding or routing tables by associating one or more Layer3 interfaces with a given VRF table. The L3 interface can be the physical source port, the VLAN or a combination of the two. Based on the input L3 interface, a VRF_ID is obtained which is used to access the VRF table.

The VRF_ID field is derived from the SOURCE_TRUNK_MAP table, VLAN table, VXLT table, or VFP table. Its value is used as part of the key in L3 entry table. VRF_ID field from the VFP table has the highest priority, next is from the SOURCE_TRUNK_MAP table, then VLAN table. For example, if the VRF_ID value is 7 in the SOURCE_TRUNK_MAP table and VRF_ID is 2 in VFP, then the value 2 will be pass to the L3 table as part of the search key. The device supports VRF for IPv4 or IPv6 of non-tunneled, tunneled, unicast, or multicast packets.

VRF_ID value can be set under the SOURCE_TRUNK_MAP table, VLAN_TABLE or VFP Table. For VFP, the VRF_ID is set under VFP_POLICY table. If the entry in the VFP is hit and its action is to assign VRF_ID, then the VRF_ID value is derived from VFP_POLICY table.

4.1.2 VLAN Resolution

For a packet from a front-panel port, the internal O-VID or I-VID, ingress VLAN tag actions, and source virtual port group are derived from the VLAN assignment schemes. The following table summarizes the VLAN assignment through the available resources.

Table 3: VLAN Assignment Schemes

VLAN Assignment Scheme (in order of highest priority first)	Input	Output
Ingress VLAN Translation	Ingress port group incoming O-VID and I-VID	Internal O-VID and I-VID, ingress VLAN tag action profile pointer, source virtual port group ID (overlaid with internal I-VID).
Subnet-based	Source IP address	Internal O-VID and I-VID, ingress VLAN tag action profile pointer.
Protocol-based	EtherType field	Internal O-VID and I-VID, ingress VLAN tag action profile pointer.
Port-based	Ingress port	Internal O-VID and I-VID, ingress VLAN tag action profile pointer.

For a HiGig packet that does not require HiGig lookup, if there is a hit in the VLAN ContentAware Processor, then the internal-O-VID, I-VID, and ingress tag actions are derived from the matching entry. Otherwise, the internal O-VID is extracted from the module header, the internal I-VID is extracted from the packet if the packet is double tagged, and the ingress outer and inner tag actions are DO_NOT_MODIFY.

4.1.3 Shared VLAN Learning

The BCM56070 supports both independent VLAN learning (IVL) and shared VLAN learning (SVL). In IVL, the device learns each MAC address independently for each VLAN. SVL is a per-device feature that can be enabled through the ING_CONFIG_64 register. When this feature is enabled, the device associates multiple VLANs to a single shared FID (forwarding database ID) through the VLAN table. All learning and forwarding is based on MAC+FID key. As a result, the switch learns each MAC address for the group of VLANs that share the same FID. Lookup for multicast packets can be programmed through ING_CONFIG_64.LOOKUP_L2MC_WITH_FID_ID. When this feature is disabled, the device operates in IVL mode of operation.

4.1.4 Private VLAN

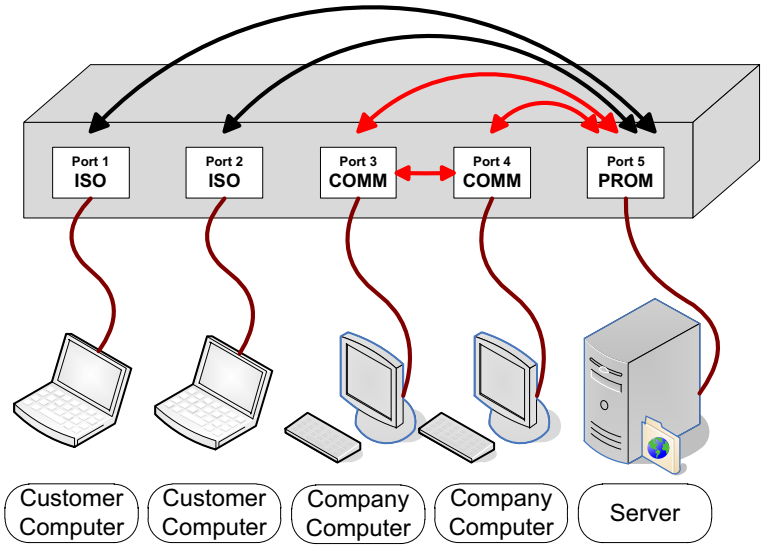
Private VLAN is a mechanism that provides additional Layer 2 traffic isolation between ports within a regular VLAN. This feature places constraints on traffic flow between specific ports in a VLAN. For instance, in an enterprise network, client ports can communicate with server ports, but not among each other.

Private VLAN is port based and it can be enabled through the PVLAN_ENABLE field in the PORT_TABLE for each port. There are three types of private VLAN ports:

- Promiscuous port – A promiscuous port can communicate with all interfaces, including the community and isolated ports within a private VLAN.

- Isolated port – An isolated port has complete Layer 2 separation from all other ports within the same private VLAN except for the promiscuous ports. Private VLANs block all traffic to isolated ports except traffic from promiscuous ports. Traffic received from an isolated port is forwarded only to promiscuous ports.
- Community port – Community ports communicate among themselves and with the promiscuous ports. These interfaces are isolated at Layer 2 from all other interfaces in other communities or isolated ports within their private VLAN.

Figure 7: Private VLAN



Private VLAN ports are associated with a set of supporting VLANs that are used to create the private VLAN structure. There are three types of private VLANs:

- Primary VLAN – The VLAN that a promiscuous port is associated with is called primary VLAN.
- Isolated VLAN – The VLAN that an isolated port is associated with is called Isolated VLAN. Each isolated VLAN must bind to a primary VLAN.
- Community VLAN – The VLAN that a community port is associated with is called community VLAN. Each community VLAN must bind to a primary VLAN.
- Isolated VLAN and Community VLAN are also called secondary VLANs.

The BCM56070 supports port based Private VLAN, providing additional Layer 2 security and traffic isolation between ports within a regular VLAN. Each port can be configured as a promiscuous, isolated or community port associating with a primary or secondary (isolated, community) VLAN. Private VLANs can be extended across multiple devices through inter-switch ports that transport primary, isolated, and community VLANs to other BCM56070 or other devices that also support private VLANs. The BCM56070 supports Private VLAN with the following specifications:

- All promiscuous ports in a single Private VLAN use only one VLAN ID.
- All isolated ports in a single Private VLAN use only one VLAN ID.
- Each group of community ports in a Private VLAN uses only one VLAN ID.
- Support for multiple community VLAN IDs in a private VLAN.
- A promiscuous port can only be associated with one primary VLAN.
- A group of isolated ports can only be associated with one secondary VLAN.
- A community port can only be associated with one secondary VLAN.
- Community and isolated ports cannot be associated with the same secondary VLAN.
- Supports multiple ports associated with a primary VID.
- Supports multiple ports associated with a secondary VID.

- A secondary VLAN can have one primary VLAN associated with it.
- Private VLAN can work across multiple switches through HiGig ports that transport PVLAN across multiple devices.
- Supports multiple private VLANs on a switch or on a switched network.
- Supports shared VLAN learning to allow separate IDs for switching and filtering.

To support private VLAN, the device uses the FID (forwarding Data Base ID) for shared VLAN learning. This allows all ports belonging to a Private VLAN to be learned in the regular L2 table using a common VLAN ID. The FID used for L2 forwarding and learning purposes for all Private VLAN ports is the VID of the promiscuous port which can be assigned from the VLAN table or VCAP to the incoming packets on isolated and community ports according to the internal OVID.

4.1.5 Double Tagging

The BCM56070 VLAN double tagging architecture provides flexibility and easy implementation for supporting a variety of applications that include provider bridging, DSLAM, and VLAN cross connect. The BCM56070 treats the incoming traffic on any of the GE ports the same. All incoming packets into the GE ports are parsed and examined for both outer and inner tags. The VLAN ContentAware Processors and the VLAN translation tables can be used for VLAN tag manipulation to provide a comprehensive Q-in-Q feature set.

Double-tagging architecture in the BCM56070 provides flexible VLAN tag manipulation for supporting the following actions:

- Insertion for outer tag if no outer tag
- Insertion for inner tag if no inner tag
- Insertion for outer tag even if outer tag exists
- Deletion of outer or inner tags
- Modifying inner and outer tags
- Modifying outer TPID
- Support for four outer TPID per port

Figure 8 and Table 4 lists terminology and acronyms used to discuss VLAN tagging in the next sections.

Figure 8: VLAN Tag Terminology

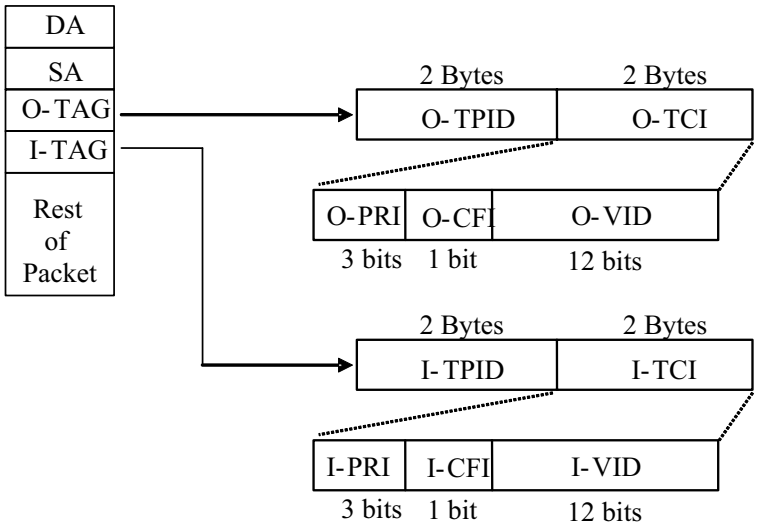


Table 4: VLAN Tag Terminology

VLAN Tag Term	Description
C-CFI	Customer CFI bit
C-PRI	Customer priority
C-TAG	Customer VLAN tag
C-TCI	Customer tag control information
C-TPID	Customer tag protocol identifier
C-VID	Customer VLAN ID
DA	Destination MAC address
DT	Double Tagged. A packet is considered double tagged if bytes 12 and 13 of the packet match any one of the ingress port's configured outer TPIDs, and bytes 16 and 17 of the packet match the per-chip configured inner TPID value.
I-CFI	Inner CFI bit
I-PRI	Inner priority
I-TAG	Inner VLAN tag
I-TCI	Inner tag control information
I-TPID	Inner tag protocol identifier
I-VID	Inner VLAN ID
O-CFI	Outer CFI bit
O-PRI	Outer priority
O-TAG	Outer VLAN tag
O-TCI	Outer tag control information
O-TPID	Outer tag protocol identifier
O-VID	Outer VLAN ID
SA	Source MAC address
S-DEI	Service VLAN drop eligible indicator
SIT	Single Inner-tagged. A packet is considered single inner-tagged if bytes 12 and 13 of the packet do not match any one of the ingress port's configured outer TPIDs but match the per-chip configured inner TPID.
SOT	Single Outer-tagged. A packet is considered single outer-tagged if bytes 12 and 13 of the packet match any one of the ingress port's configured outer TPIDs, and bytes 16 and 17 of the packet do not match the per-chip configured inner TPID.
S-PRI	Service VLAN priority
S-TAG	Service VLAN tag
S-TCI	Service tag control information
S-TPID	Service tag protocol identifier
S-VID	Service VLAN ID
UT	Untagged. A packet is considered untagged if bytes 12 and 13 of the packet do not match any one of the ingress port's configured outer TPIDs and do not match the per-chip configured inner TPID.

NOTE:

- Outer tag is determined by comparing outer TPID with the allowed outer TPIDs of the ingress port, defined by PORT.OUTER_TPID_ENABLE. The OUTER_TPID_ENABLE contains 4 bits, indicating which of 4 TPIDs stored in the ING_OUTER_TPID table are allowed outer TPID. If there is a match with any of the allowed outer TPIDs, the packet is considered to be outer tagged.
- Inner tag is determined by comparing inner TPID with VLAN_CTRL.INNER_TPID. If they match, the packet is considered to be inner tagged.

4.2 Packet Flow

The following section describes how packets flow from various ingress and egress ports. At a general level, this section illustrates how the device treats packet flow and congestion.

4.2.1 L2 Ingress Packet on Ethernet Ports

The device supports standard Ethernet or IEEE 802.3 packets (64 bytes to 1522 bytes), as well as jumbo packets up to 12 KB. The packet parser within the ingress block parses an incoming packet and extracts all the fields needed by the ingress logic and ContentAware processor. Next, the device determines if it is tagged, untagged, or priority tagged. If it is untagged or priority tagged, the device assigns the VLAN ID from the following tables:

- Rule-based table (VCAP)
- VLAN_XLATE table
- Subnet-based table
- Protocol-based table
- Port-based table (default)

If the packet is tagged, then VLAN ID is picked up from the packet itself. The packet then passes through a VLAN lookup.

4.2.2 L3 (IPv4 Unicast) Ingress Packet on Ethernet Ports

The following describes how the device treats the L3 packet upon being received at the switched network ports.

In addition to supporting L2 switching, the device also supports L3 packet switching, or routing. The purpose of L3 switching is to allow a packet from one VLAN to communicate with another VLAN. Unlike L2 switching, L3 packet switching is based on the packet destination IP address (DIP), not the MAC_DA address. For the device to do L3 switching, the L3_Enable bit must be set.

4.2.3 Ingress and Egress Packet CPU Port

As previously mentioned, the CPU port is part of the CMIC block. The device connects to an external CPU through a PCIe interface. The device allows users to send and receive packets from the CPU port to any of the GbE or HiGig ports. The DMA engine works with the interfaces to support packet data transfer to and from the CPU. On the PCI bus, the DMA engine operates as a PCI bus master for this purpose. The CMIC is able to send two types of packets to the CPU: Ethernet packets and HiGig format packets.

4.3 L2 Feature List and Operations

This section describes the Layer 2 features the BCM56070 device supports.

In addition to the L2 features described in this section, the BCM56070 device also supports L2 switching. The primary component of this feature is the L2 table, which is used for the following three applications:

- VLAN-based bridging (basic) for switching L2 unicast packets.
- Single-VLAN cross-connect for DSLAM where forwarding is done based on the outer vlan_id.
- Double-VLAN cross-connect for DSLAM where forwarding is done based on the outer and inner vlan_id.

This table is managed by hardware to allow line-rate (wire speed) switching for all packet sizes and conditions. Only regular VLAN and VFI-based entries can be learned, but all entry types can be aged.

To support VLAN-based bridging, learning and forwarding is done based on the MAC address and outer VLAN ID (VID). This provides for seamless support of Independent VLAN Learning (IVL). For switching decisions, the {DST_MAC and VID} are used to search the L2 table. When a match is found, the entry provides module ID and port ID, TGID, VPG, or L3 indication. When the address is not found, the packet is a Destination Lookup Failure (DLF) and is flooded to the entire VLAN. Broadcast packets (MAC_DA == FF:FF:FF:FF:FF:FF), are also flooded (or broadcast) to the entire VLAN.

To support Single VLAN cross-connect application, forwarding is done based on the outer VLAN ID. When a match is found, the entry provides module ID and port ID, TGID, VPG, or L3 indication. When a match is not found, the packet is dropped and it is optionally copied to CPU.

To support Double VLAN cross-connect application, forwarding is done based on the outer and inner VLAN IDs in the form of {OVID, IVID}. When a match is found, the entry provides module ID and port ID, TGID, VPG, or L3 indication. When a match is not found, the packet is dropped and it is optionally copied to CPU.

To support VFI-based bridging, learning and forwarding is done based on the MAC address and VFI. Instead of using module ID and physical ports, learning and forwarding is based on a Virtual Port basis. For switching decisions, the {VFI and DST_MAC} are used to search the L2 table. When a match is found, the entry provides destination virtual port. When the address is not found, the packet is a Destination Lookup Failure (DLF) and is flooded to the entire VFI. Broadcast packets (MAC_DA == FF:FF:FF:FF:FF:FF), are also flooded (or broadcast) to the entire VFI.

4.4 L3 Feature List and Operations

This section describes the Layer 3 features the BCM56070 supports.

4.4.1 Layer 3 (IPv4 Routing)

The device supports line rate routing of IPv4 packets for all packet sizes and conditions. Inside the L3 logic, the first stage is the source IP address lookup (SIP). If the SIP is not found, then the device does nothing. If it is a match, then the device updates the L3 hit bit and performs a Destination address lookup (DIP).

4.4.2 IP Multicast

The switch supports IPv4 and IPv6 multicast applications, such as multimedia conferences, real video and real audio, and push technology. Such applications depend heavily on single point to multipoint delivery of service. These and other next generation applications require a robust multicast routing protocol and efficient delivery of multicast packets without adversely affecting standard unicast IP routing.

Some of the IP protocols accepted and deployed by the Internet community include: Distance Vector Multicast Routing Protocol (DVMRP), Protocol Independent Multicast Dense Mode (PIM-DM), Protocol Independent Multicast Sparse Mode (PIM-SM), and Multicast Extensions to OSPF.

The device supports the following industry-standard IP multicast protocols:

- DVMRP
- PIM-DM
- PIM-SM
- Protocol Independent Multicast Source Specific Mode (PIM-SSM)
- Multicast Open Shortest Path First (MOSPF)

4.4.2.1 Equal-Cost Multipath

Equal-cost multipath (ECMP) is a technique for routing packets along multiple paths of equal cost. If multiple equal-cost routes to the same destination exist, ECMP can be used to provide load balancing among the redundant paths. As the forwarding logic has several next hops for any given destination, it must use some method to choose which next hop should be used for a given data packet. Various routing protocols, including Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (ISIS) explicitly allow ECMP routing, and some routers also allow equal cost multipath usage with RIP and other routing protocols.

The device provides ECMP support that allows the packet to be forwarded to an ECMP group. A group can have at most 64 paths. A path within a group is selected based on a CRC32 hash of the following types:

- {UDF, TCP_Dst_Port, TCP_Src_port, 0, SIP}
- {UDF, TCP_Dst_Port, TCP_Src_port, DIP, SIP}
- RTAG7

4.5 Tunneling Features

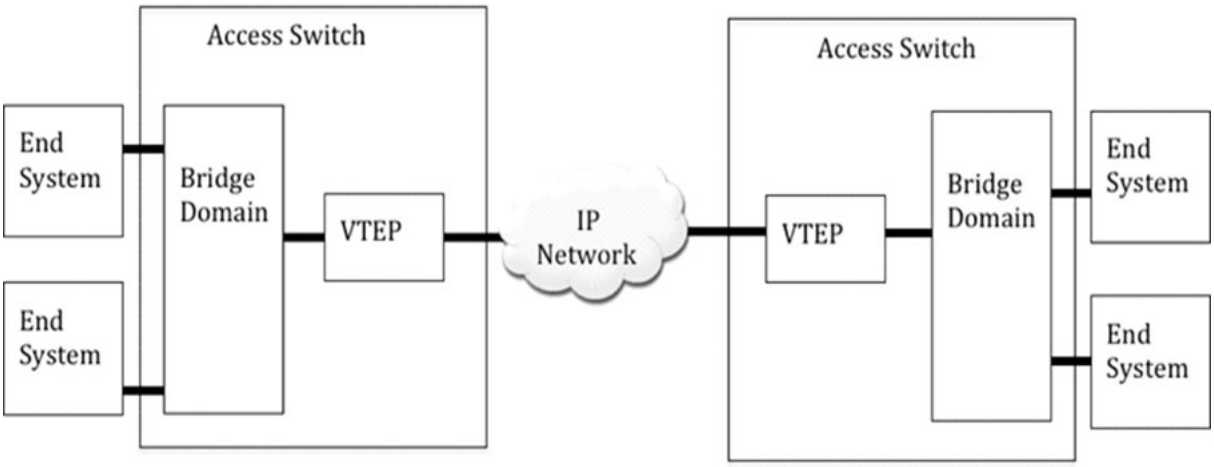
4.5.1 VXLAN

Virtual eXtensible Local Area Network (VXLAN) is a framework to address the need for overlaying networks over within virtualized data centers accommodating multiple tenants. VXLAN is a tunneling scheme to overlay Layer 2 network on top of Layer 3 network, called *VXLAN overlay network*. Each overlay is also known as a VXLAN segment.

The VTEP is defined as the VXLAN Virtual Tunnel End Point. VTEPs are responsible for encapsulating the Virtual Machine (VM) traffic in a VXLAN header as well as stripping it off and presenting the destination virtual machine with the original L2 packet. Thus, the VXLAN Network Identifier (VNID) identifying a overlay network and VXLAN-related tunnel or outer header encapsulation are processed only by a VTEP device. The virtual machine never sees it.

The BCM56070 is positioned for deployment in potential VTEP devices, with the goal to provide point-to-point VTEP functionality between non-VXLAN aware wireless or wired networks and VXLAN aware networks. Or, between two VXLAN segments.

Figure 9: VXLAN System Deployment Overview



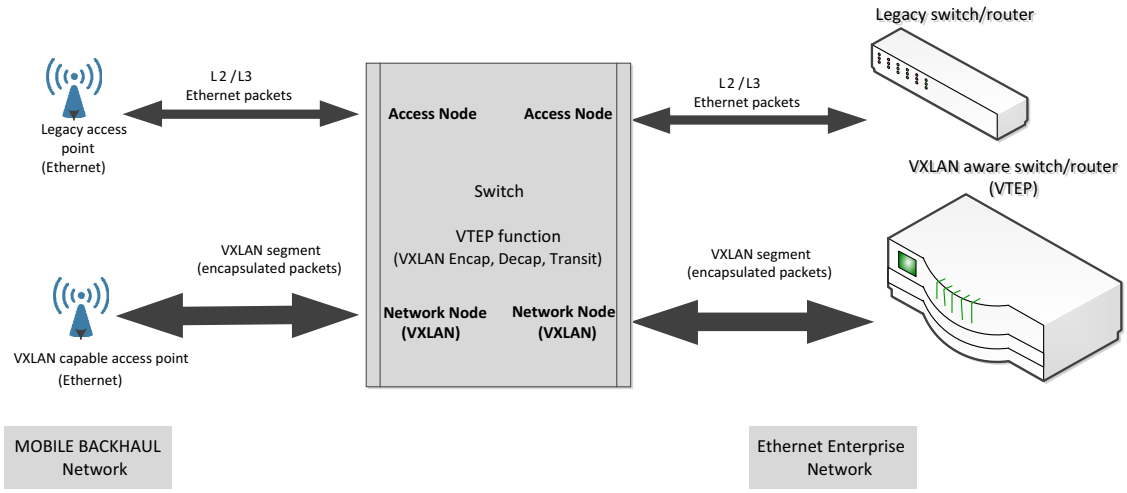
4.5.1.1 VXLAN Deployments

This section describes the network deployments of BCM56070 that requires VXLAN support.

Figure 10 illustrates a typical VXLAN deployment with the BCM56070 supporting wireless traffic from wireless access points (APs) to wired devices like servers, switches, and routers in a point-to-point network system configuration. The BCM56070 receives and transmits a mix of VXLAN and non-VXLAN Ethernet packets between these devices.

- Traffic flow between access nodes.
  - Non-VXLAN packets are treated as native Ethernet packets.
  - BCM56070 will perform normal L2 bridging and L3 routing.
- Traffic flow between network nodes (VXLAN-aware network).
  - BCM56070 performs the transit function for the pass-through of VXLAN traffic.
  - BCM56070 alternatively performs VXLAN DECAP-ENCAP for traffic between two VTEPS.
- Traffic between access nodes and network nodes.
  - BCM56070 performs VTEP function to encapsulate into VXLAN packet for upstream traffic.
  - BCM56070 performs VTEP function to decapsulate VXLAN packet for downstream traffic.

Figure 10: Network System Deployment Overview



4.5.1.2 VXLAN Requirements

The main requirement for VXLAN is to provide VTEP functionality by providing tunnel termination and generation services on behalf of an end-point that is not VXLAN-aware. That is, VXLAN provides point-to-point L2 Ethernet payload tunneling over a VXLAN tunnel. The primary VXLAN requirements are as follows:

- In the upstream direction, to add a VXLAN tunnel header to unicast, multicast, and broadcast L2 packets.
- In the downstream direction, to terminate outer VXLAN tunnel header and forward the inner unicast, multicast, and broadcast L2 (Ethernet) payload.
- To support at least 1024 VNI flows (VXLAN entries for encapsulation, decapsulation, and DECAP-ENCAP).

NOTE: VXLAN implementation in BCM56070 is limited to provisioned environment that does not need any hardware learning support.

The BCM56070 implements the following functions to support VXLAN:

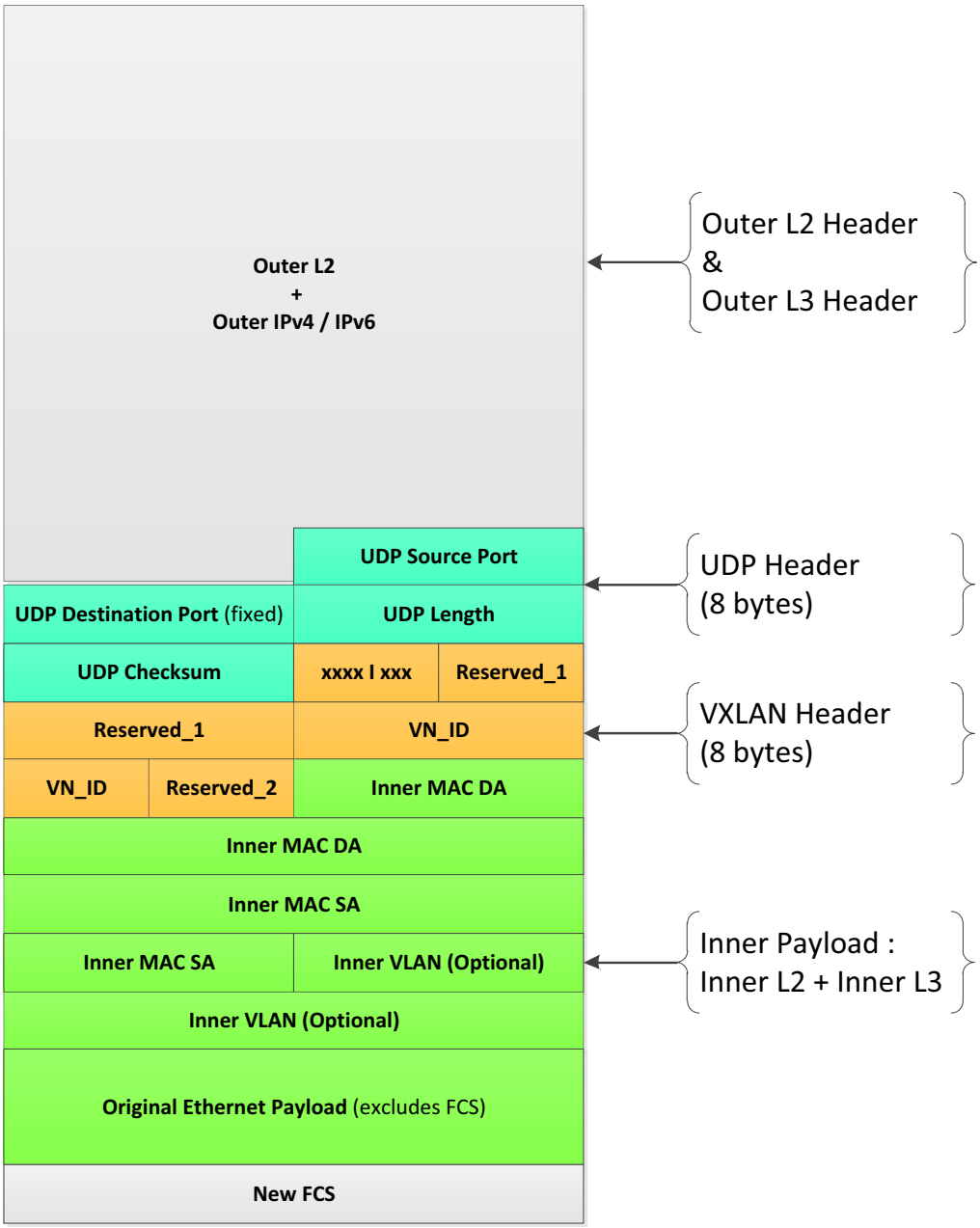
- Support overlay endpoint function for VXLAN (VTEP) – Ethernet payloads
- Support transit function between VTEP or VXLAN aware networks – Ethernet and IP payloads
- Support DECAP-ENCAP between VTEP or VXLAN aware networks – Ethernet payloads
- Support outer IPv4 and IPv6 unicast and multicast VXLAN tunnels
- Support mapping of multiple VN_IDs to one tunnel
- Flexibility to detect reserved and undefined header fields in future VXLAN RFC and forward to CPU
- Support load balancing based on UDP header
- Support overlaying a hash on the UDP source port (between restricted range)
- UDP destination port configurable and UDP source port range from 49152 to 65535
- UDP checksum – Receiving VXLAN packets (DECAP), *must* accept packets with UDP checksum = 0, and if checksum is non-zero, then may discard the packet or ignore checksum
- UDP checksum – Sending VXLAN packets (ENCAP) with UDP checksum = 0
- Support load balancing on regular L3 and L4 fields hashing in transit flows
- Statistics – Per tenant counters during encapsulation and decapsulation, and per-tunnel counters and per-next-hop egress counters during encapsulation
- IGMP snooping for VXLAN terminated packets (post decap)
- Per port enable for VXLAN

- VXLAN packet MAC learning
- Option to drop L2 payload coming downstream (DECAP), if it has a VLAN tag
- Support P2P (E-Line) service and MP2MP (E-LAN) service for VXLAN.
  - E-Line does P2P switching without MAC_DA lookup and IFP based VXLAN flow can support P2P (E-Line) services based on VNID, port, or both
  - E-LAN requires MAC-DA look up and L2 Entry based VXLAN flow supports E-LAN services using MAC-DA, VNID

4.5.1.3 VXLAN Packet Format

The VXLAN architecture is based on VXLAN packet format and header as defined in the RFC.

Figure 11: VXLAN Packet Overview



4.5.1.3.1 VXLAN Packet Headers

The VXLAN packet has an inner MAC frame with its own Ethernet header with source and destination MAC addresses along with the Ethernet type, plus an optional VLAN. The inner Ethernet MAC frame is encapsulated with the following four headers:

VXLAN Header + Outer UDP Header: + Outer IP Header + Outer Ethernet Header

The outer destination MAC address is generally used as the address of the target VTEP or of an intermediate Layer 3 router. The outer VLAN tag is optional. If present, it may be used for delineating VXLAN traffic on the LAN.

4.5.1.3.1.1 VXLAN Header

This is an 8-byte field that has the following components:

- Flags (8 bits): where the I flag *must* be set to 1 for a valid VXLAN Network ID (VNI). The other 7 bits (designated *R*) are reserved fields and *must* be set to zero on transmission and ignored on receipt. It is expected that future RFC versions may assign some values.
- VXLAN Segment ID or VXLAN Network Identifier (VNI): this is a 24-bit value used to designate the individual VXLAN overlay network on which the communicating VMs are situated. VM in different VNIs cannot communicate with each other.
- Reserved fields (24 bits + 8 bits): *Must* be set to zero on transmission and ignored on receipt. It is expected future RFC versions may assign some values.

4.5.1.3.1.2 Outer UDP Header

This is the outer UDP header with a source port provided by the VTEP and the destination port being a well-known UDP port.

- Destination Port: IANA has assigned the value 4789 for the VXLAN UDP port. This value *should* be used by default as the destination UDP port. Some early implementations of VXLAN have used other values for the destination port, and to enable interoperability with older implementations, the destination port *should* be configurable.
- Source Port: The UDP source port number should be calculated using a hash of fields from the inner packet.

  Example: Hash of the inner Ethernet frame's headers. This is to enable a level of entropy for load balancing of the VM-to-VM traffic across the VXLAN overlay. When calculating the UDP source port number in this manner, it is *recommended* that the value be in the dynamic or private port range from 49152 to 65535.
- UDP Checksum: It *should* be transmitted as zero. When a packet is received with a UDP checksum of zero, it *must* be accepted for decapsulation.

4.5.1.3.1.3 Outer IP Header

This is the outer IP header of initiating VTEP. The destination IP address can be a unicast or multicast IP address. When it is a unicast IP address, it represents the IP address of the terminating VTEP.

4.5.1.3.1.4 Outer L2 Header

The outer destination MAC address may be the address of the target VTEP or of an intermediate Layer 3 router. The outer VLAN tag is optional. If present, it may be used for delineating VXLAN traffic on the LAN.

4.5.1.4 VXLAN Network Flows

The BCM56070 supports the following VXLAN packet flows:

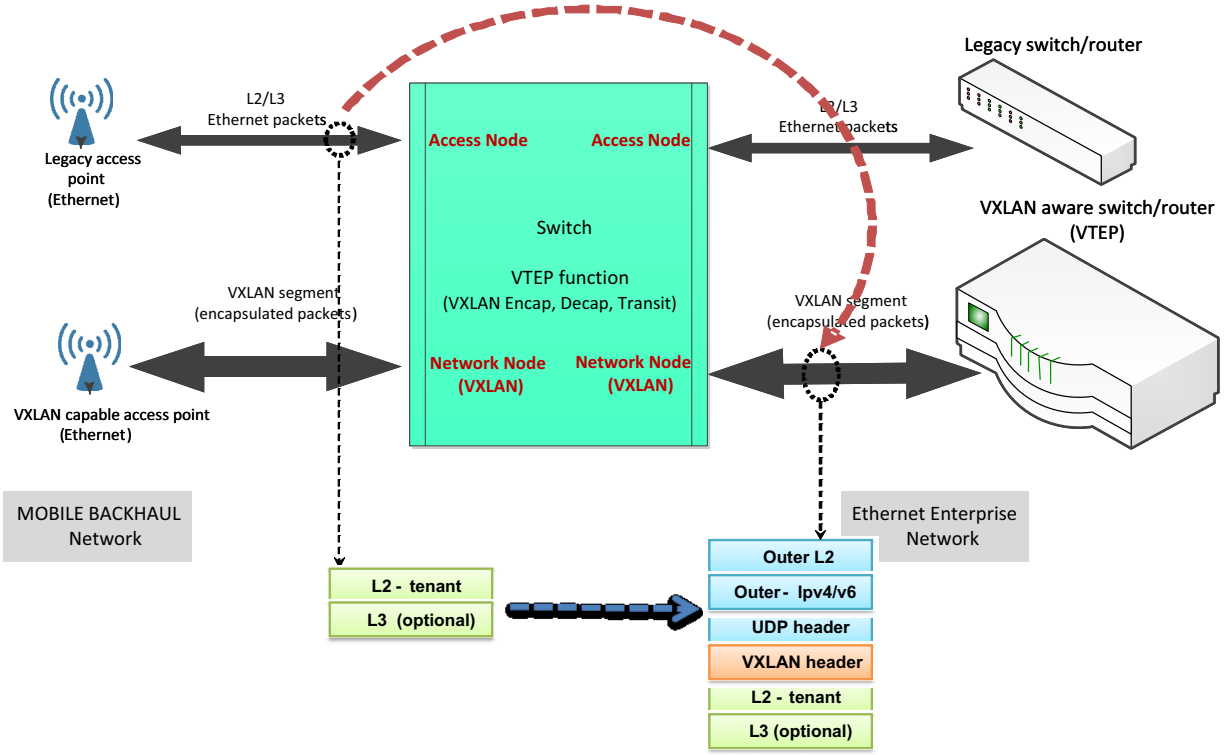
- ENCAP – Ethernet payload flow
- DECAP – Ethernet payload flow
- DECAP-ENCAP – Ethernet payload flow
- Transit (pass-through) – VXLAN payloads for load balancing

4.5.1.4.1 ENCAP

Access node (local network port) ↔ Network node (VXLAN segment)

- Ingress ↔ L2 (Ethernet packet) tenant packet identified for doing ENCAP
- Egress ↔ VXLAN encapsulated Ethernet packet

Figure 12: Encap Network System Flow



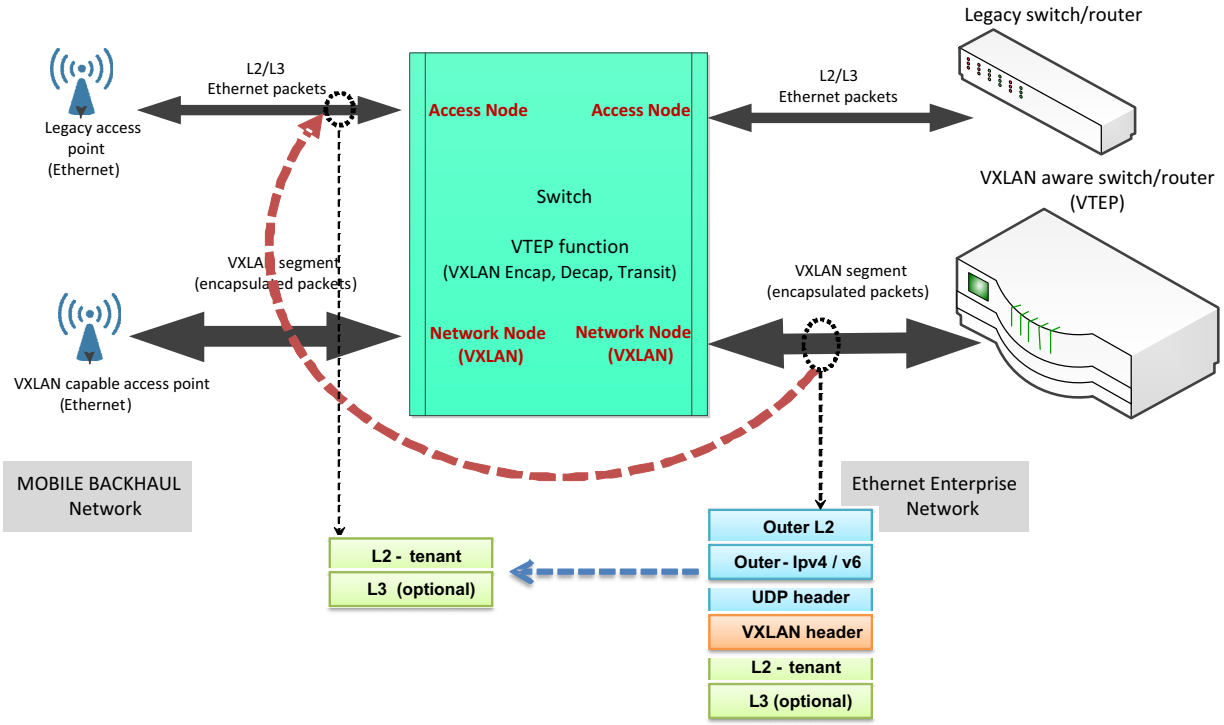
4.5.1.4.2 DECAP

Network node (VXLAN segment) ↔ Access node (local port)

- Ingress ↔ VXLAN packet identified for DECAP
- Egress ↔ Inner payload (L2 packet)

Traffic from a VXLAN tunnel is decapsulated and forwarded to a locally connected tenant. The BCM56070 supports L2 payload packet processing out of VXLAN tunnel, with no L3 routing.

Figure 13: DECAP Network System Flow



4.5.1.4.3 DECAP–ENCAP

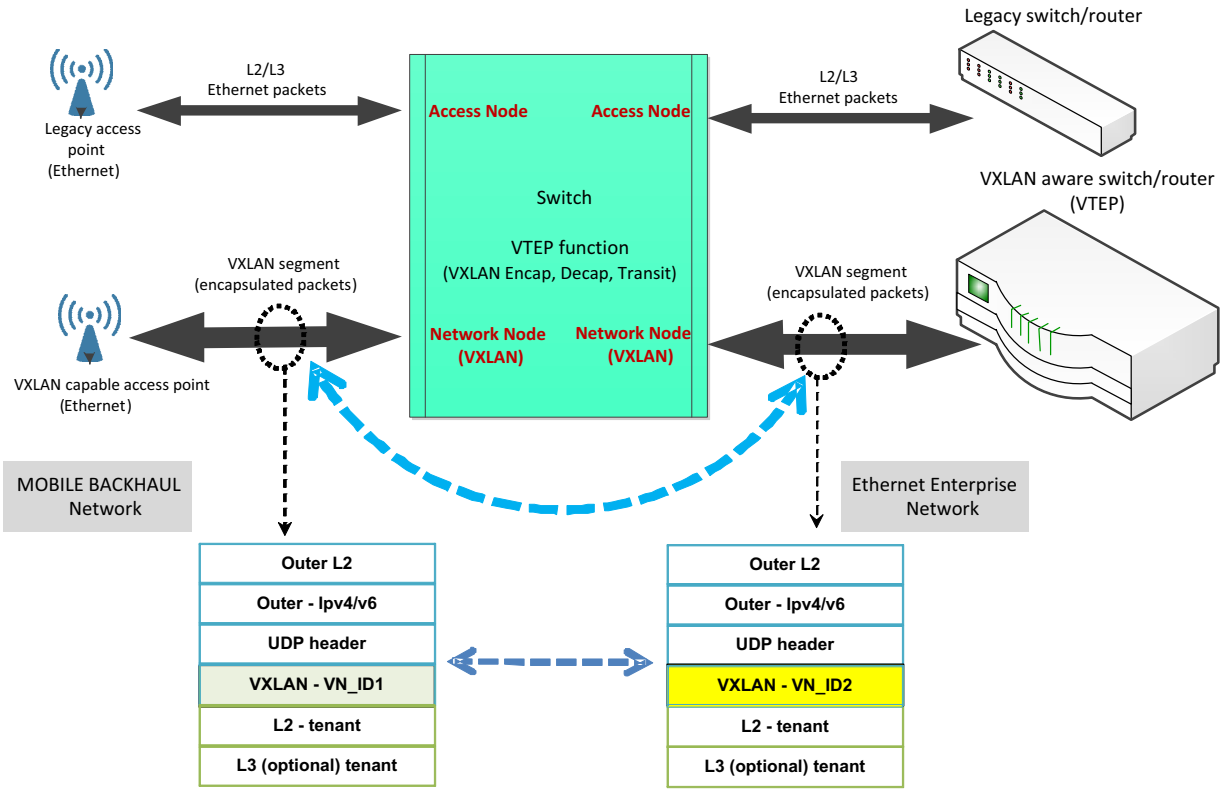
Network node (VXLAN1 segment) ↔ Network node (VXLAN2 segment)

- Ingress ↔ VXLAN1
- Egress ↔ VXLAN2

This requires terminating VXLAN1, doing inner MAC-DA lookup and then encapsulating VXLAN-2.

BCM56070 deployment for DECAP-ENCAP is a light version of service sharing between VXLAN networks, in a point-to-point flow.

Figure 14: DECAP–ENCAP Network System Flow



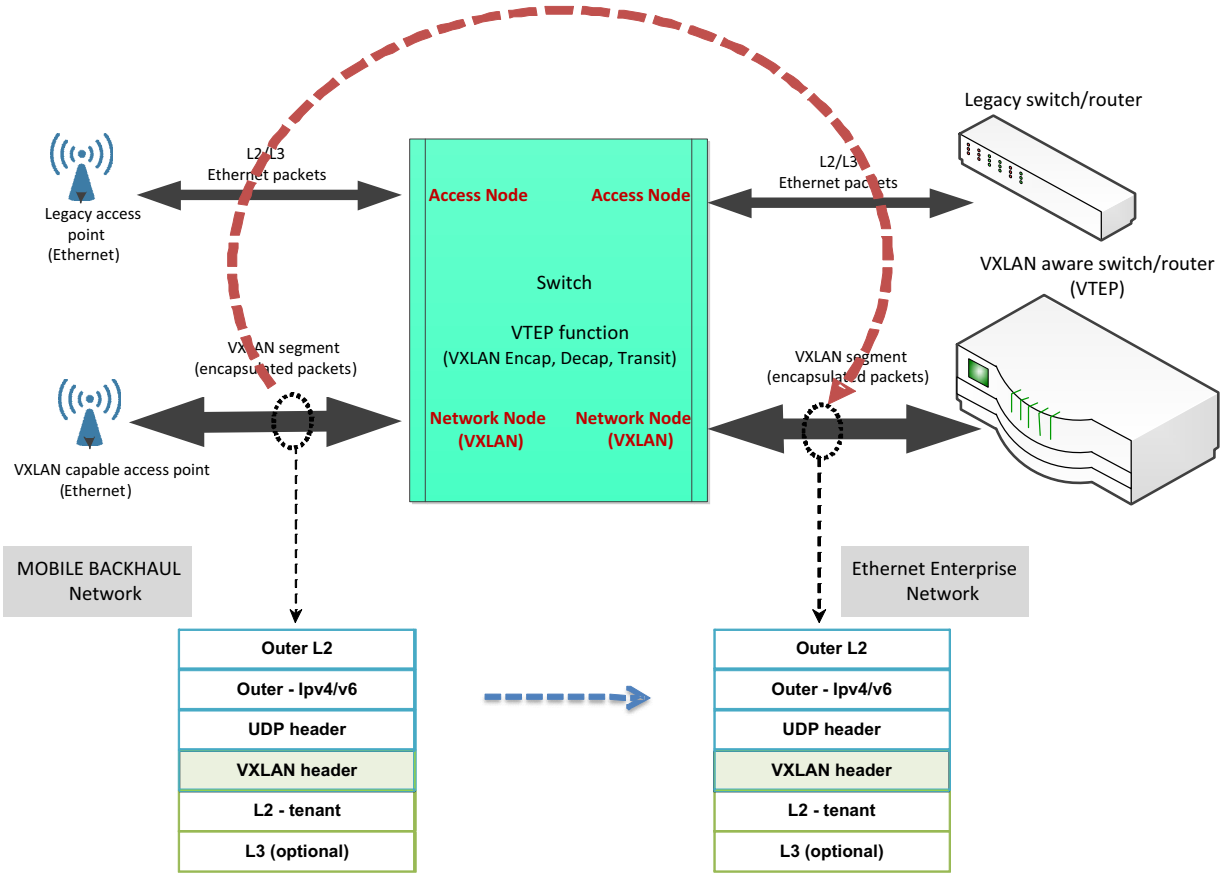
4.5.1.4.4 TRANSIT

Network node (VXLAN1 segment) ↔ network node (VXLAN1 segment)

- Ingress ↔ VXLAN1 encapsulated
- Egress ↔ VXLAN1 encapsulated (unmodified)

Standard L3 routing and VXLAN is not terminated.

Figure 15: TRANSIT Network System Flow



4.6 Maximum Transmission Unit Checking

To discard packets considered to be oversized in the network, the BCM56070 includes configurable Maximum Transmission Unit (MTU) thresholds per type of packet flow.

4.7 Port Extender

Bridge Port Extender is also known as IEEE8.21BR, and the standard “specifies the devices, protocols, procedures and managed objects necessary to extend a bridge and its management beyond its physical enclosure using 802 LAN technologies” (refer to https://1.ieee802.org/dcb/802-1br/).

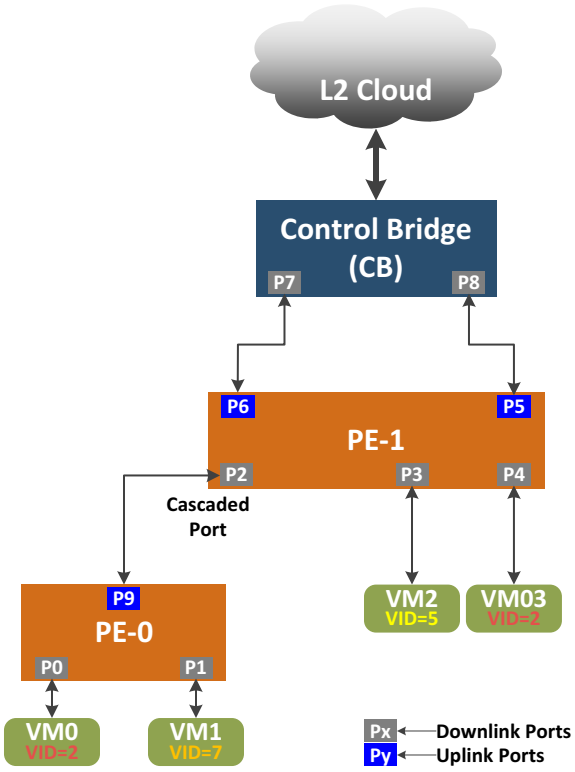
This section describes how the BCM56070 device supports PE functions. The following figure depicts a basic PE based Network with a single Controlling Bridge (CB) and two Port Extenders: PE-0 and PE-1.

NOTE: The BCM56070 device does not support controlling bridge functions.

PE-0 functions as an Access PE, while PE-1 can function as both a Transit PE (with respect to VM0 and VM1 traffic) and an Access PE (with respect to VM2 and VM3 traffic).

Any port facing the direction of the VMs (traffic moves away from the CB; ports shaded gray) are called downlink ports. Any port facing the direction of the CB (traffic moves toward the CB; ports shaded cobalt-blue) are called uplink port. The downlink port (P2) connecting the transit PE-1 to the access PE-0 is referred to a cascaded port. Any packet going out of this port is expected to have an ETAG.

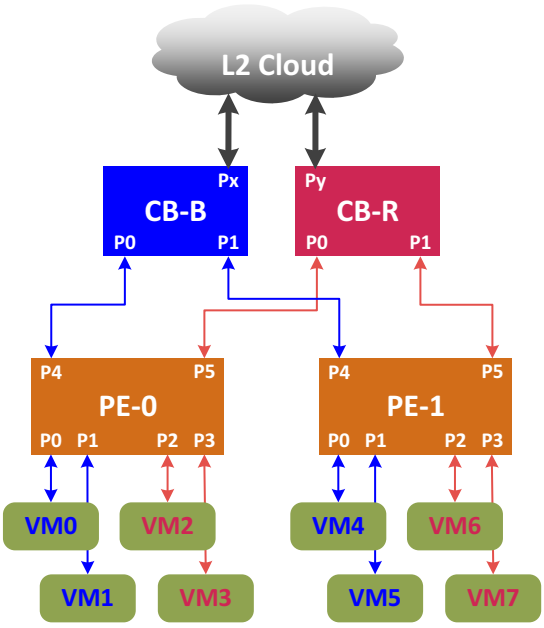
Figure 16: Basic Port Extender (PE) Based Network



4.7.1 Port Extender Access Network

Figure 17 shows a port extender (PE)-based access network with two controlling bridges (CB-R and CB-B) and two port extenders (PE-0 and PE-1). Both PEs have four attached VMs. In this configuration, the downlink ports P0 and P1 of both PEs have a designated uplink port (P4), and ports P2 and P3 of both PEs have a designated uplink port (P5). Multicast and unicast traffic coming from VM0, VM1, VM4, and VM5 are forwarded to CB-B. Similarly, traffic coming from VM2, VM3, VM6, and VM7 are forwarded to CB-R. PE downlink ports add a special ingress port-based tag, ETAG, on the ingress upstream packet to uniquely identify the source VM of the packet.

Figure 17: PE Access Network



CB-B and CB-R support packet forwarding between associated VMs in the network. CB-B switches packets between VM0, VM1, VM4, and VM5, and CB-R switches packets between VM2, VM3, VM6, and VM7. Each CB learns the association between associated VMs and their L2 addresses {MAC_SA, VLAN}. Each CB also does L2 forwarding based on L2 addresses {MAC_DA, VLAN} between their physical ports P0 and P1. For every outgoing packet, the CB attaches an ETAG based on the destination VM to which the packet is sent. Each CB port also defines a unique namespace for associated VMs, thus limiting the scope of the ETAG's VID to specific CB ports of in the network. CB-B and CB-R have two namespaces, each defined by ports P0 and P1. Traffic between the CB-B and CB-R domains is forwarded through the L2 cloud.

PE-0 and PE-1 forward the downstream packet arriving at ingress ports P4 and P5 to the correct downlink ports based on the ETAG that identifies the destination VM (or associated VMs for multicast). Each PE-0 and PE-1 ingress uplink port has a forwarding database that associates the VM (or VMs for multicast) to downlink ports. Port extenders forward downstream packets based on this database, which is preconfigured by CB-B and CBR. The forwarding databases are managed by the Port Extender Control Protocol (PECP) control plane running between CB and PE.

In the event of a failure in one of the CBs (for example, CB-B), the CB-R can quickly become the active CB of the network by changing the default uplink association of the PE downlink ports P0 and P1 from P4 to P5. This ensures a very fast network failover.

The port extender architecture simplifies traffic forwarding for all intermediate layers, which are composed of PEs instead of L2-switches, between the access switch and the VM. The PE also simplifies failover and subsequent convergence of the access network. Because the CB receives all the traffic from end hosts, and there is no local switching at the servers or any intermediate layers, the CB can become the network administration point for applying policies. All intermediate PEs are required to support Ethernet congestion management protocols such as PAUSE or PFC. This makes the underlying Ethernet infrastructure more resilient to congestion. The PEs also apply Dot1P-based QoS assignments to packets so that specified classes of traffic are ensured lower latency and minimal drop performance throughout the layers.

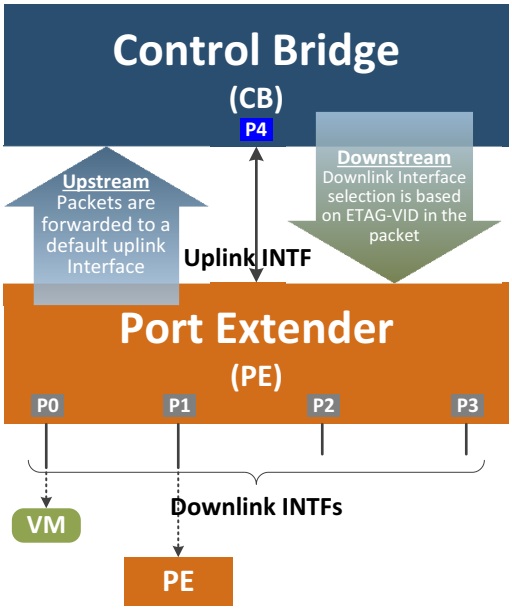
4.7.2 Port Extender

Port Extender (PE) provides a data path to and from end hosts to a CB. In a typical data center network, a hypervisor L2 switch, or blade-server L2 switch acts as a PE, and the CB is a Top of Rack (ToR) access switch. The simple and configurable forwarding model of a PE enables quick and easy management of PEs compared to traditional L2 switches.

PEs support the following two interface types:

- Uplink interfaces providing connectivity toward the associated CB.
- Downlink interfaces providing connectivity toward end hosts (or network access points).

Figure 18: PE Interface Types



Packets to and from PEs have an 8-byte tag called an ETAG, which has a unique EtherType. Packet flow to the CB from the end devices is referred to as upstream traffic flow, and packet flow from the CB to the end hosts is referred to as downstream traffic flow.

For upstream traffic, the PE selects destination uplink interfaces based on the ingress downlink interface through which the packet was admitted. For downstream traffic, the PE selects destination downlink interfaces based on the ingress uplink interface through which the packet was admitted. The ETAG's VID field identifies the associated VM or virtual switch interface (VSI). The PEs are responsible for dot1P-based filtering and scheduling of upstream and downstream traffic, and dot1p-based shaping for upstream traffic.

The PEs support L2 congestion management mechanisms such as Ethernet-PAUSE or PFC to reduce packet loss and head-of-line blocking in the access network.

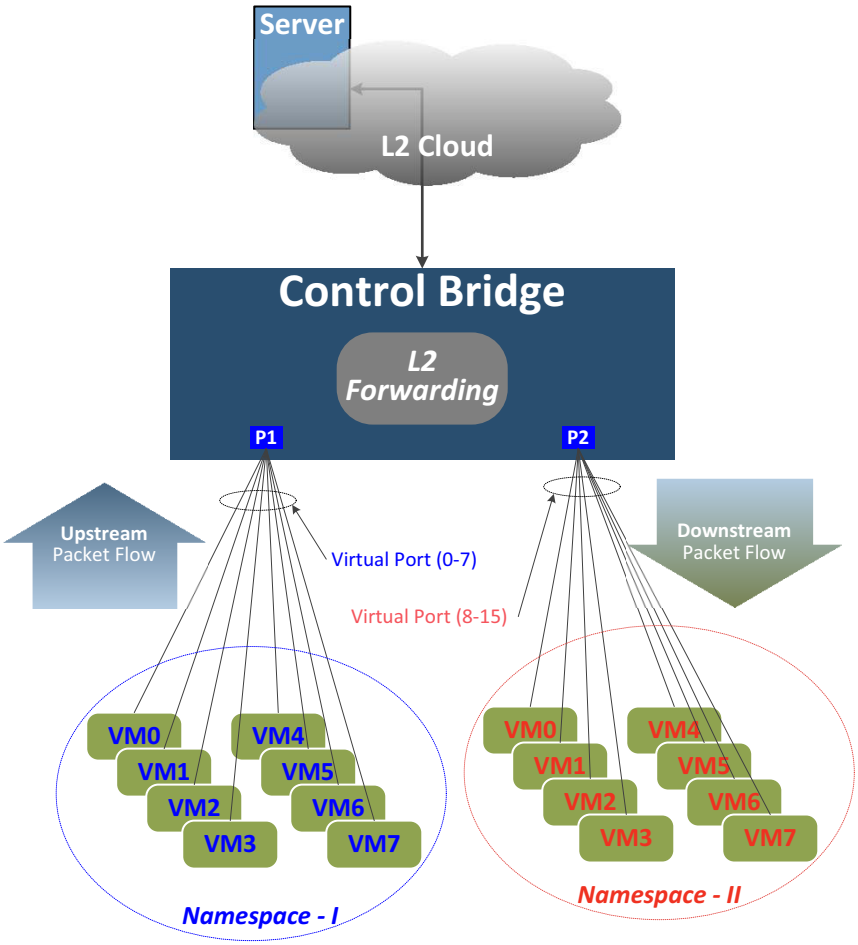
4.7.3 Controlling Bridge

NOTE: To be complete and informative, this document describes the role and functions of both the Port Extender (PE) and Controller Bridge (CB). BCM56070 family, however, supports only the PE functions.

Each Controlling Bridge (CB) configures and manages its associated PEs. The CB establishes a management and control plane with each of its PEs. It also sets up each PE for traffic forwarding and creates multicast and unicast forwarding tables in each PE through the port extender control protocol (PECP) running between the PE and the CB.

Each port of a CB can be a namespace for all the PEs and end hosts connected to that port. Each end station uses a unique VSI to represent itself in traffic flows between itself and the CB through the ETAG's VID. The CB maps each ETAG VID and associated physical port to a virtual port. The CB supports L2 forwarding between virtual ports and between virtual and physical ports. The CB is also responsible for registering MAC addresses associated with PEs for forwarding traffic to the PEs. A CB can only be actively connected to a VM through one physical downlink port at a time, while other paths can be used for redundant backups.

Figure 19: Controlling Bridge



The CB provides accessibility, upstream packet rate, upstream burst rate, QoS, and congestion management across all the PEs (or end stations) and network access points in the network.

4.7.4 Packet Format for PE Access Network

The Port Extender framing rules are listed below:

- All frames interchanged between PEs and an associated CB carry the ETAG. The CB and attached PEs learn each other's capabilities through Link Layer Discovery Protocol (LLDP) exchanges using untagged Ethernet frames. If there is a match in capabilities, they switch to port extender operation mode.
- All frames with ETAGs can optionally carry an STAG or CTAG.
- If an ETAG is present, it is located after the IEEE 802.3 source address and prior to the STAG or CTAG.
- Frames exchanged at the MAC layer, such as the IEEE 802.3 MAC control frames used for PAUSE and priority-based flow control, are transmitted without an ETAG or IEEE 802.1Q tag. This is consistent with existing MAC control mechanisms.
- ETAG frames can be as small as 64 bytes or it can be as large as the network allows. All network elements are allowed. Like to the IEEE 802.1ad standard, these include the overhead of both the ETAG and the STAG or CTAG.

The Port Extender packet format is shown in the following figure.

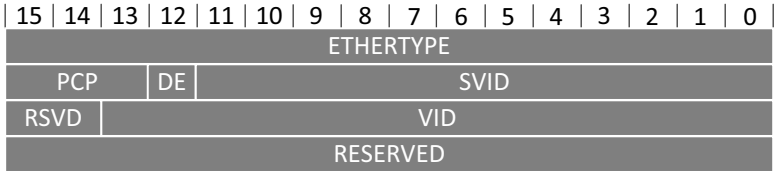
Figure 20: Port Extender Packet Format

4.7.5 ETAG Header Format

The ETAG header format has the following characteristics:

- EtherType [16] defines the IEEE 802.3 type field used to determine if a frame is carrying an ETAG.
- Extended Port VLAN ID (VID) identifies the downlink interface (and hence the VM and VSI) associated with this frame. For upstream packets, the VID indicates the source VM and VSI. For downstream packets, the VID identifies the destination VM and VSI. VID values below 4096 are for unicast destinations, and the values between 4096 and 16383 are used for multicast replication trees. The BCM56070 supports only the Base Port Extender version of the IEEE 802.1BR specification. This means the BCM56070 supports only 4K unicast channels and 12k multicast channels. Total support is up to 16K channels.
- Source Extended Port VLAN ID (SVID) is valid for downstream packets only, and if the source VM and VSI, and the destination VM and VSI are in the same namespace, it identifies the VM and VSI where the packet originated. Otherwise, the SVID is populated with zero. The SVID is used by the access PE to perform multicast source pruning of downstream traffic when SVID != 0.
- Priority Code Point and Discard Eligibility (PCP and DE) are the QoS attributes of the frame.

Figure 21: ETAG Header Format



4.7.6 Port Extender Control and Management Protocol

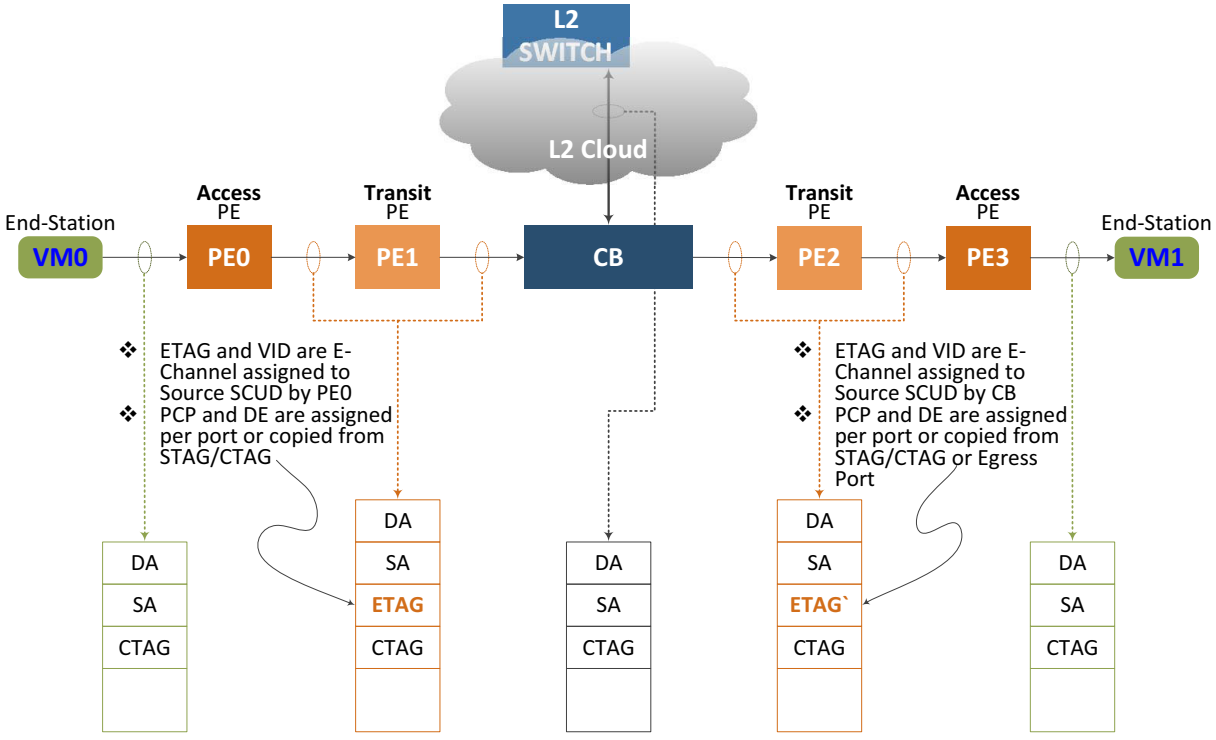
The Port Extender Control and Management Protocol runs between a Controlling Bridge (CB) and a PE to bring up or bring down attached PEs. It provides association to the VM and VSI and establishes multicast groups across the VSIs. A single instance of the protocol runs between a PE and its CB on each link or link bundle that connects them. The protocol namespace is unique for each instance of the CB. Once a link between the PE and the CB comes up, the two entities negotiate capabilities through a Data Center Bridging Exchange (DCBX) and pick up the highest supported version and control channel VID (generally VID = 1) to be used to exchange messages on the link. Once the control channel VID is up and running on the link, all further DCBX messages are exchanged over the control channel VID.

4.7.7 Port Extender Network-Level Packet Flows

4.7.7.1 L2 Unicast Packet Flow

The following figure shows how a unicast packet travels from VM0 to VM1 through a PE-based access network. PE-0 and PE-1 carry traffic upstream from VM0 to the CB. PE-1 is a downlink port, because connecting to another PE is considered a cascaded port. The CB is the central network policy management entity and performs L2 traffic forwarding. PE-2 and PE-3 carry traffic downstream with PE-2's downlink port as a cascaded port. PE-1 and PE-2 are called transit-IVs and PE-0 and PE-3 are called access-IVs. The access PE assigns ETAGs based on the ingress port. The transit port extenders are not expected to assign ETAGs.

Figure 22: Unicast Packet Flow (Network)



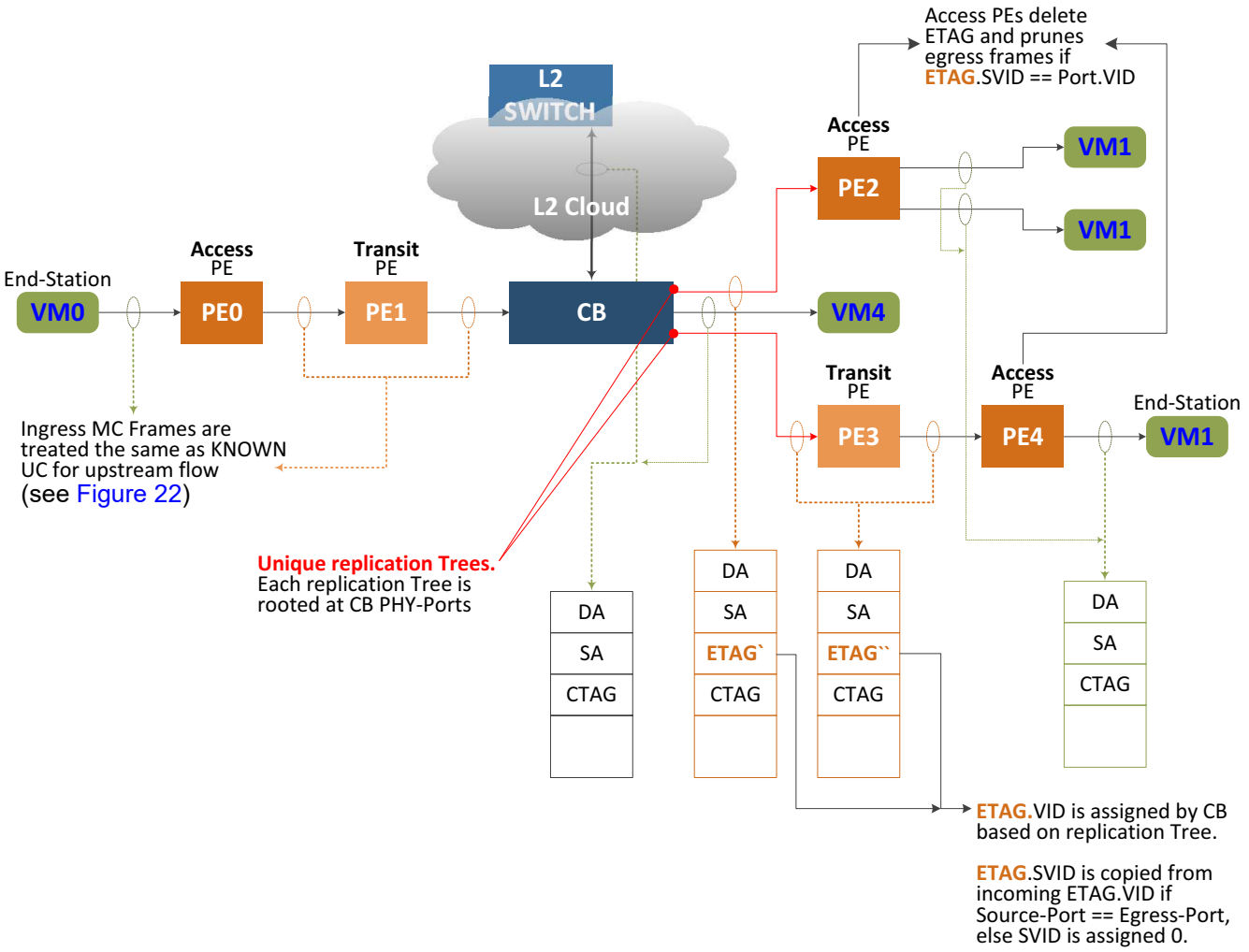
As the unicast packet travels from VM0 to VM1 through a PE-based access network, the following events occur:

- PE-0 is responsible for assigning an ingress port-based ETAG to the packet. The ETAG's VID identifies the packet's source VM. PE-0 also populates the PCP and DE fields of the ETAG from the packet's STAG/CTAG, or based on the ingress port. The S-VID field is set to zero. Packets ingressing with ETAG VID == 0 are treated as non-ETAG tagged packets and are assigned an ingress port-based ETAG. However, the incoming packet's PCP and DE values are retained.
- PE-0 forwards traffic received on the downlink port to a preconfigured uplink port. The packet does not undergo L2 lookup or learning.
- PE-1 expects all incoming packets at a downlink port to include an ETAG tag. The ETAG's SVID field is not looked up by PE-1.
- PE-1 forwards traffic received on the downlink port to a pre-configured uplink port. The packet does not undergo L2 lookup or learning.
- When the CB receives the packet from PE-1, it uses {ingress port, ETAG.VID} to identify which VM the packet originated from (in this case, VM0). Any policies for traffic from VM0 are applied. The CB also learns the association between {MACSA, VLAN} and {ingress port, ETAG.VID}.
- The CB performs L2 forwarding lookups on the packet's {MACDA, VLAN}. The result can be either a destination that is reachable through the L2 cloud, or it can be a local VM.
  - If the destination is reached through the L2 cloud, then the forwarding lookup result is an egress port. The ETAG is deleted and the packet is sent to the L2 cloud.
  - If the destination is a local VM, then the forwarding lookup results in {egress port, VID}.
- If the egress port is different from the ingress port, then the packet goes to a different namespace domain. The ETAG.SVID field is set to zero. The ETAG.VID field is assigned from the forwarding lookup.
- PE-2 expects all packets ingressing at the uplink port to be ETAG tagged. PE-2 drops all non-ETAG tagged packets and copies them to the CPU. PE-2 also checks whether the ETAG format is correct for downstream packet flow.
- PE-2 forwards the packet based on {ingress port, ETAG.VID} lookup that results in a destination port (the downlink port to PE-3). The ingress port in the key identifies the namespace (CB port) for the VID. The packet is forwarded to PE-3.
- PE-3 also forwards the packet based on {ingress port, ETAG.VID} lookup. Since the packet is going to a VM (and not to another PE), PE-3 removes the ETAG tag from the packet before sending it to VM1.

4.7.7.2 L2 Multicast Packet Flow

This subsection describes how a multicast packet travels from VM0 to VM1 through VM4. In the following figure, PE-0 and PE-1 carry traffic from VM0 upstream to the CB. The downstream traffic is carried by PE-2, PE-3, and PE- 4 to end stations VM1, VM2, and VM3. VM4 is directly connected to the CB.

Figure 23: Multicast Packet Flow (Network)



In the upstream direction, all packets are first sent to the CB, regardless of unicast or multicast. Each downlink port will forward traffic to their associated uplink ports. CB does forwarding lookups based on {MACDA, VLAN} and determines the recipients for each packet. In the downstream direction, PEs are capable of doing multicast replication based on ETAG.VID (VID values from 4096 to 16383 are reserved for Multicast IDs). Therefore, the CB sends only one copy of the packet to each PE connected to it. Each downlink PE represents a single multicast replication tree with a unique 14-bit multicast replication pointer.

As a multicast packet travels from VM0 to VM1 through VM4, the following events occur:

- When a multicast packet is received at PE-0 from VM0, packet processing is identical to the unicast case. An ETAG tag is inserted and the packet is forwarded to a preprogrammed uplink port.
- Processing at PE-1 is also identical to the unicast case. Packets are forwarded to a preprogrammed uplink port.
- When the CB receives the packet from PE-1, it uses {ingress port, ETAG.VID} to identify the VM where the packet originated (VM0 in this case). Any policies for traffic from VM0 are applied. The CB also learns the association between {MACSA, VLAN} and {ingress port, ETAG.VID}
- The CB performs L2 forwarding lookups on the packet's {MACDA, VLAN}. The result is either a destination reachable through the L2 cloud or a downlink port.
  - For recipients reachable through the L2 cloud, the ETAG tag is deleted and the packet is sent to the cloud. This type of traffic includes traffic to the L2 switch and VM4.
  - On each downlink port, if the packet has VM receivers behind the PE, the CB sends one copy of the packet with the ETAG's VID field set to the multicast distribution tree identifier.
- For packets going back out the ingress port, the ETAG tag's SVID field is populated from the incoming packet's ETAG.VID. Otherwise, the ETAG.SVID is set to 0.
- When PE-2 and PE-3 receive the packet from the CB, they perform a lookup on {ingress port, ETAG.VID} to get a set of downstream ports. PE-2 and PE-3 replicate the packets to each of the downstream ports.
- For packets egressing on ports connected to VMs (for example, PE2 and PE4), the ETAG is deleted. For these ports, the PEs also check whether the packet originated from the same port (that is, if ETAG.SVID != 0 and ETAG.SVID = Port.VID). If the packet originated from the same port, the packet is not forwarded.
- For packets egressing on ports connected to another PE (that is, cascaded ports), the ETAG is passed through.

4.7.8 Port Extender with Spanning Tree

The extended bridge (CB + PEs) acts as single dot1Q bridge. The CB runs the spanning tree protocol for the extended bridge. Because the extended bridge represents a loop-free forwarding tree, the PEs do not run spanning tree protocol and are not expected to generate or consume BPDUs. PEs forward BPDUs like normal packets with ETAGs to uplinks and downlinks. For L2 switches interconnected through PEs, the CB runs Per-VLAN Spanning Tree Protocol instances like any other L2 switch. The BPDUs are generated by the switch CPU to non-PE links as well as PE links. BPDUs sent to PE ports include an ETAG.

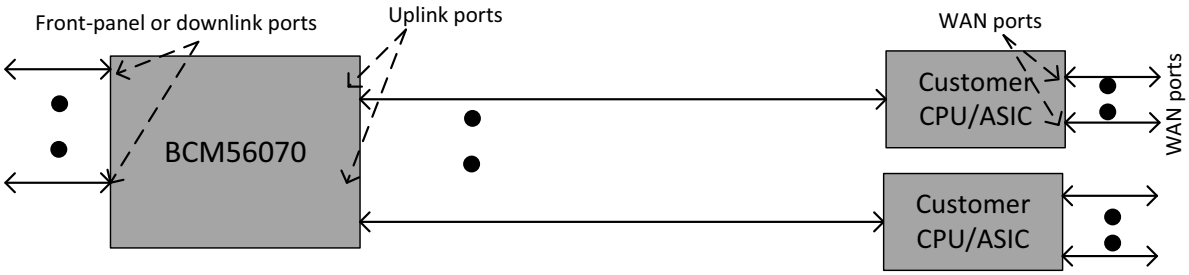
4.8 Mac-in-Mac-Lite

4.8.1 Overview

Mac-in-Mac-Lite (MiML) is used to optionally forward Ethernet packets with unknown SAs received on a BCM56070 port to the upstream customer CPU or ASIC with MiML encapsulation. In that application, packets with unknown SA are forwarded to the customer CPU or ASIC to learn the source MAC address. Another application of MiML is to forward control frames received in the BCM56070 with a reserved MAC as a destination address to the customer CPU or ASIC to centralize control plane processing. A third application of MiML is to select some L2, L3, or L4 flows through IFP rules and forward those to the customer CPU or ASIC with an MiML header for further processing. Finally, the customer CPU or ASIC may send an MiML packet in the downstream direction to a BCM56070 with an MiML header that may identify the QoS treatment that the packet will get in the switch. In some of those applications, the MiML header can be constructed to uniquely represent the source port of the BCM56070 on which the Ethernet packet was received or the destination port of BCM56070 on which the packet will be transmitted.

An MiML port may carry Non-MiML packets. The BCM56070 device supports all other switch features including normal Ethernet (non-MiML) forwarding, mirroring, lag, scheduling, and so on, in an MiML deployment. The following figure shows the general application of MiML.

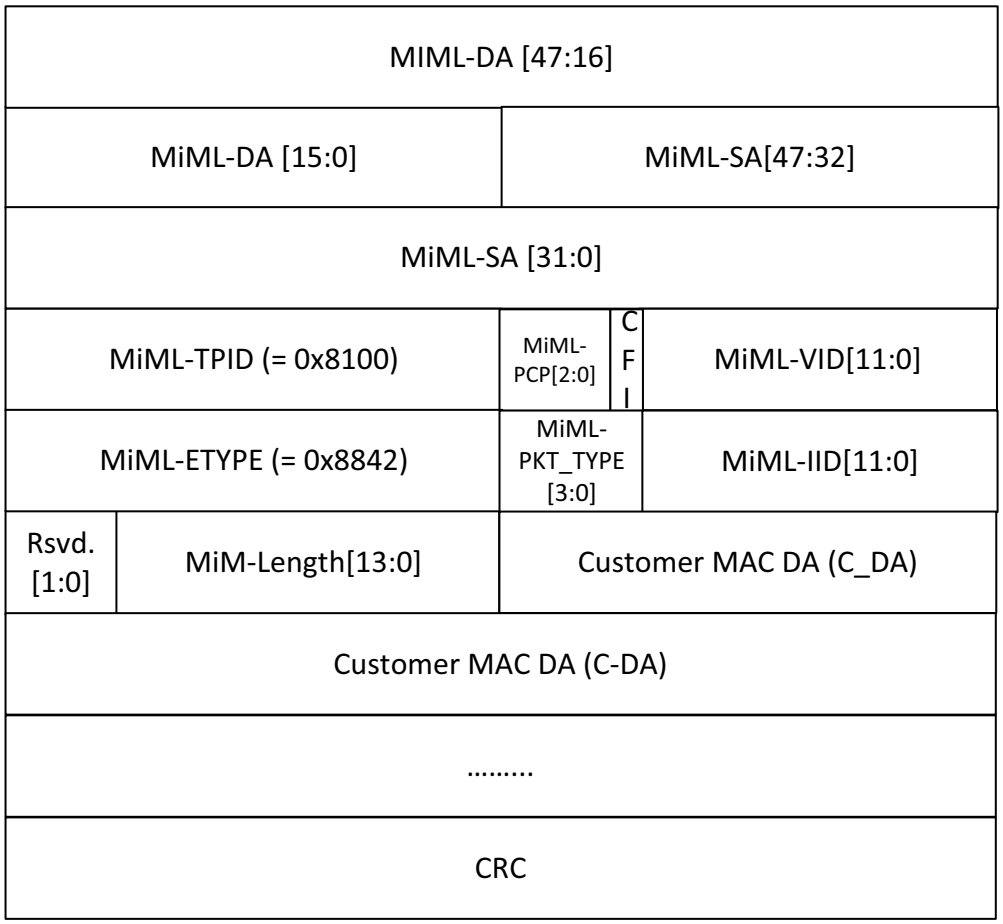
Figure 24: MiML General Application



4.8.2 MiML Packet Format

MiML uses a similar packet format as standard Mac-in-Mac (see Figure 25). However, some of the packet header fields are different.

Figure 25: MiML Packet Format

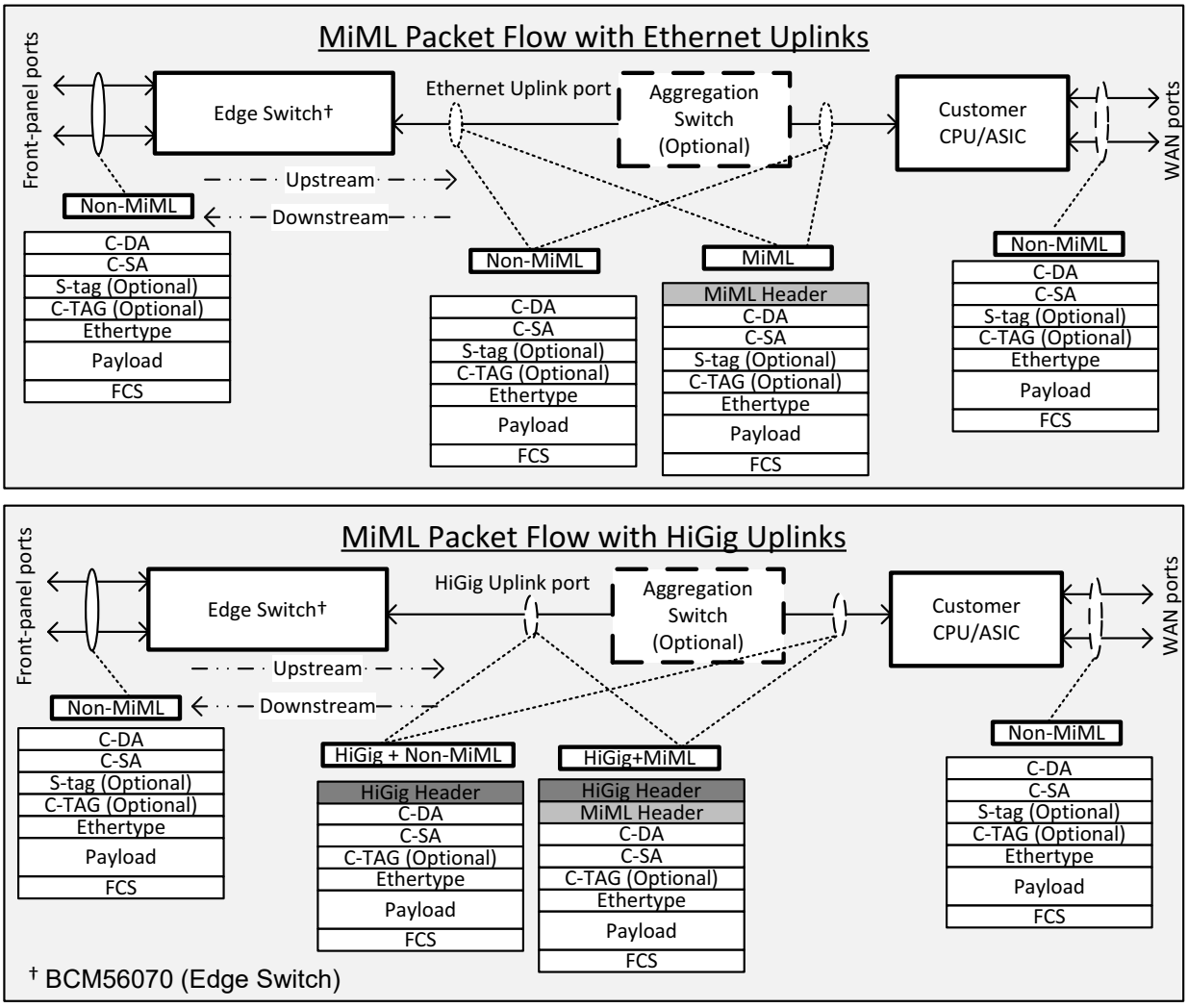


MiML_DA = B-DA
MiML_SA = B-SA
MiML_TPID = .1ad Ethertype
MIML_PCP = .1ad B-Tag PCP
MiML_CFI = .1ad B-Tag DP
MIML_VID = BVID
MiML-ETYPE = .1ah Ethertype
MiML_PKT_TYPE[3:0] = {.1ah I-IP, I-DEI}
MiML_IID[11:0] = {.1ah NCA, Reserved, ISID[24:16]}
MiML_Length[13:0] = .1ah ISID[13:0]
Note: MiML_Length can be set to 0

4.8.3 MiML Traffic Flows

The following figure illustrates the packet formats at various interfaces in systems supporting MiML with and without HiGig interfaces. As illustrated, the BCM56070 downlink ports will always receive non-MiML standard Ethernet packets. Some of those packets will be encapsulated in an MiML frame and sent to the customer CPU or ASIC through the optional aggregation switch. If an interface connecting the aggregation switch or the customer CPU or ASIC is configured as HiGig, then a HiGig header will be added on top of MiML encapsulation. Local switching among downlink ports is permitted in a MiML network, although it has not been explicitly shown.

Figure 26: MiML Traffic Flows



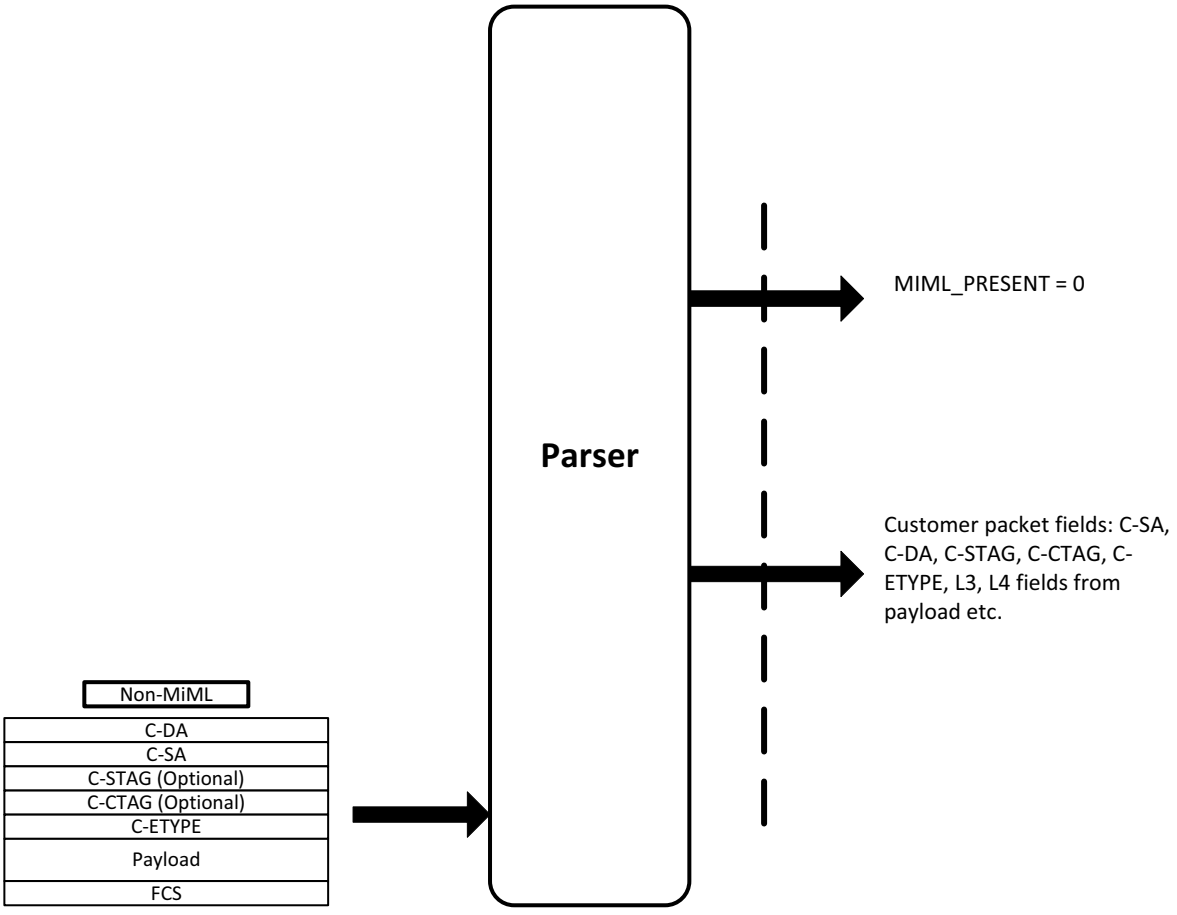
4.8.4 MiML Upstream Processing

Figure 27 and Figure 28 illustrate how various functional blocks in the BCM56070 process MiML packets in the upstream direction. Other functional blocks in the BCM56070 will process upstream packets similar to the way downstream packets or non-MiML packets are processed. In the upstream direction, one or more of the BCM56070 downlink ports will receive only standard non-MiML Ethernet packets. The packets may be locally switched to other downlink ports or may be forwarded to an uplink ports depending on switch configurations. Optionally, the packet may be encapsulated in a MiML packet depending on switch configuration. If an upstream packet (MiML or non-MiML) is forwarded on a HiGig uplink port, it will have an appropriate HiGig header (HiGig2).

4.8.4.1 Upstream Parser Functions

The following figure illustrates parser functions for upstream traffic in an MiML network. In an application, the `MIML_ENABLE` configuration bit for all downlink ports will be set to 0 by software. Also, downlink ports will not receive any HiGig packets. Therefore, the parser will treat all packets as non-MiML Ethernet packets without HiGig encapsulation and, the `MIML_PRESENT` flag from the parser will always be 0.

Figure 27: Upstream Parser Functions



4.8.4.2 Upstream VLAN Assignment

This section describes only packet processing in the upstream direction. In that direction all packets will be treated as non-MiML by the parser. The only special software configuration required is in the usage of the `ClassID` attribute of the assigned VLAN. Some of the assigned VLAN will be marked by software by using one of the associated `ClassID` bits as untrusted VLANs. In those applications, IFP will be configured to forward upstream packets received in an untrusted VLAN and with `Unresolved_SA` to be encapsulated in an MiML packet and forwarded to an uplink port.

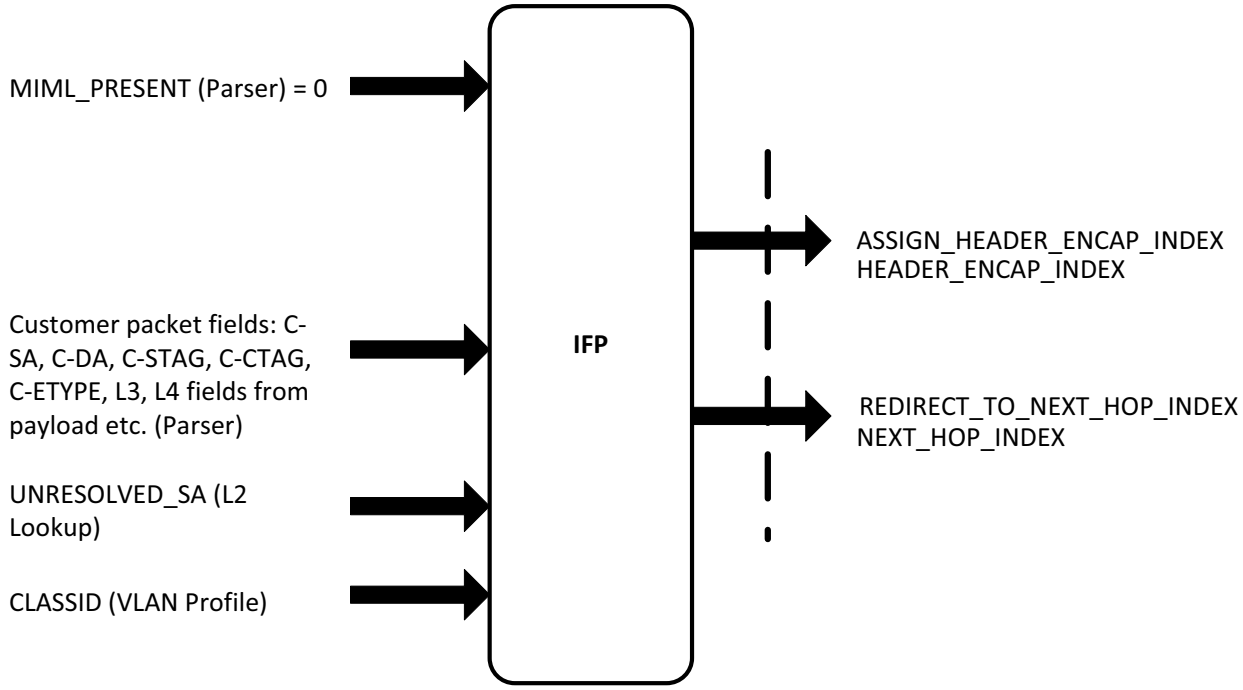
4.8.4.2.1 Upstream QoS Assignment

In the upstream direction, all packets will be identified as non-MiML. Therefore, the internal priority and color assignments are done through normal non-MiML processing.

4.8.4.2.2 Upstream IFP Processing

The following figure illustrates upstream IFP processing. Any upstream packet can be encapsulated in an MiML packet through IFP actions. If a packet is encapsulated in a MiML packet, the IFP will be configured to override its destination to redirect it to a port connected to customer CPU or ASIC for further processing.

Figure 28: Upstream IFP Processing



In an application, an upstream packet may be MiML encapsulated if it belongs to an untrusted VLAN and has an unresolved SA. A VLAN is configured as untrusted by setting the ClassID field as described in the Upstream VLAN Assignment section. The function is the same as the one discussed in the Downstream IFP Processing section.

Another reason an upstream packet may be MiML encapsulated is when it has a source MAC address that is unresolved. When set, the Unresolved_SA hardware flag from the L2 table to IFP identifies a packet for which there is no matching {VLAN, Source MAC address (or, SA)} pair on the same ingress port the packet was received. The mismatch may occur if either the {VLAN, SA} pair is not in the L2 table or if the pair is in the table but a station move has occurred that has caused the packet to be received on a different ingress port of the BCM56070. In either case, the Unresolved_SA flag in the IFP key can be optionally used to re-direct such a packet back to the customer CPU or ASIC for more processing (for example, software learning).

In addition, an upstream packet may be MiML encapsulated because of its type. For example, because it is a BPDU frame or it belongs to a certain TCP or UDP flow. Control frames like BPDU may be forwarded to the customer CPU or ASIC for centralized control plane implementation. Otherwise, a TCP or UDP flow may be identified by configuring IFP rules with L2, L3, or L4 fields in the key and frames in those specific flows may be forwarded to a customer CPU or ASIC for specialized application processing.

An IFP lookup of any upstream packet may result in an IFP action `ASSIGN_HEADEDR_ENCAP_INDEX` to encapsulate the packet in a MiML packet. The data associated with the action is `HEADER_ENCAP_INDEX` that points to a table with some fields in the MiML header. The table is shared between MiML and custom header applications. The remaining fields in a MiML header are overlayed on the `Next_Hop` table and hence Layer3 processing (routing, IP multicast) cannot be used in conjunction with MiML. The `REDIRECT_TO_NEXT_HOP_INDEX` action from the IFP along with the `NEXT_HOP_INDEX` associated with that action are used by the Packet Modification function to get those remaining MiML header fields. The limitation is acceptable in a MiML application because a packet is encapsulated in a MiML packet for further processing in the customer CPU or ASIC that supports routing and IP multicast functions. The IFP action is used by the Packet Modification function.

4.8.4.2.3 Upstream Packet Modification

In an application the upstream packet modification function will process only Ethernet packets received on downlink ports.

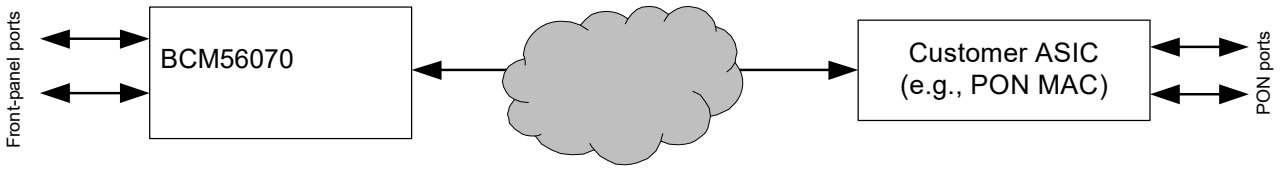
4.9 Custom Packet Header

4.9.1 Overview

Custom packet header will be used between a BCM56070 and a Customer CPU or ASIC. A custom header is a 32-bit word that is inserted after the MAC SA field in an Ethernet packet. Figure 29 illustrates a custom packet header deployment. In some deployments, there may be a Layer 2 transport network between a BCM56070 and the CPU or ASIC, and the custom header may be constructed such that packets with custom headers are transported transparently as standard Ethernet frames.

One application of a custom packet header is to forwarded Ethernet packets from the BCM56070 in the upstream direction with source port information to the upstream customer CPU or ASIC for processing, and to forward downstream frames to the destination port mapped from the custom header fields. In that application forwarding, as well as most security and QoS decisions, are made in the customer's CPU or ASIC. A second application of custom packet header is end-to-end flow control between the BCM56070 and CPU or ASIC. In that application, the BCM56070 may forward PFC frames with custom headers to the CPU or ASIC to pause traffic in the downstream direction. Similarly, the CPU or ASIC may send PFC frames with custom headers to the BCM56070 to pause traffic in the upstream direction. Appropriate custom headers are used in those applications to transport PFC frames to either the CPU or ASIC or the BCM560700 and identify which port and queues must be flow controlled.

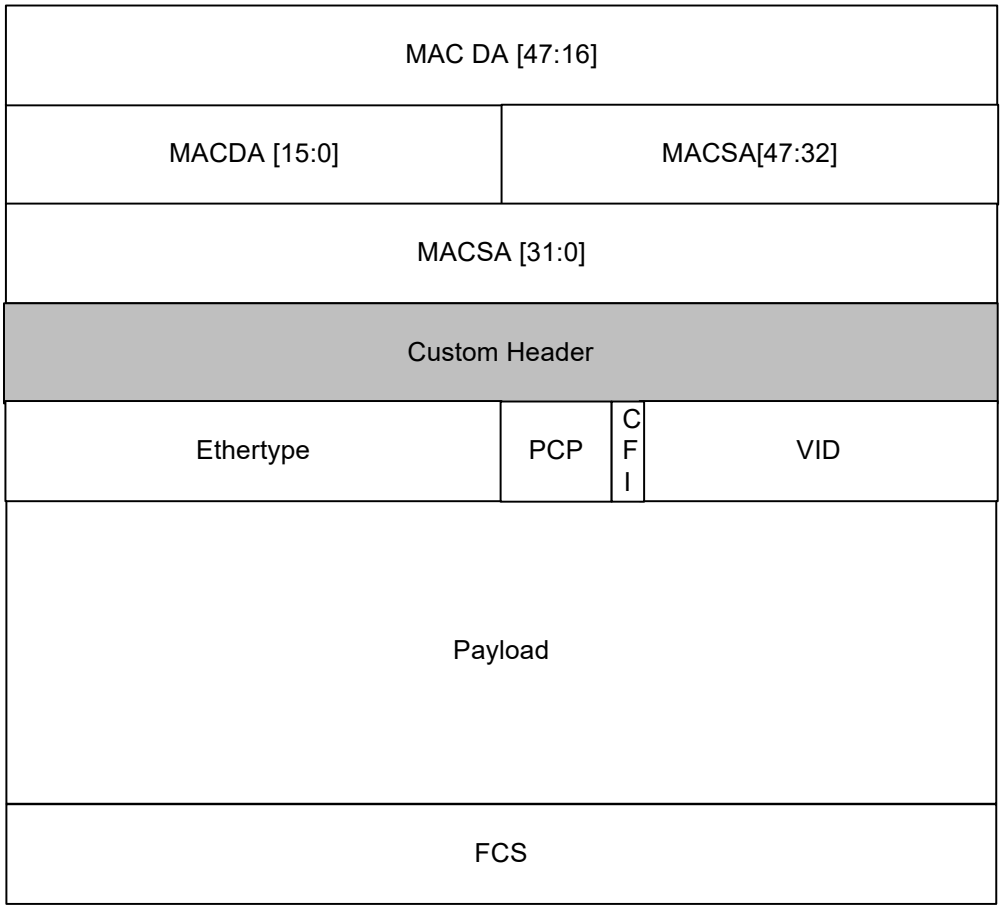
Figure 29: Custom Packet Header Deployment



4.9.2 Custom Header Packet Format

The following figure illustrates the format of packets with 32-bit custom packet header. The specific custom packet header format depends on specific applications. In some applications the custom packet header may have an EtherType field to help forward it in a transport network.

Figure 30: Custom Packet Header Packet Format



Ethertype = 0x8808 for PAUSE/PFC

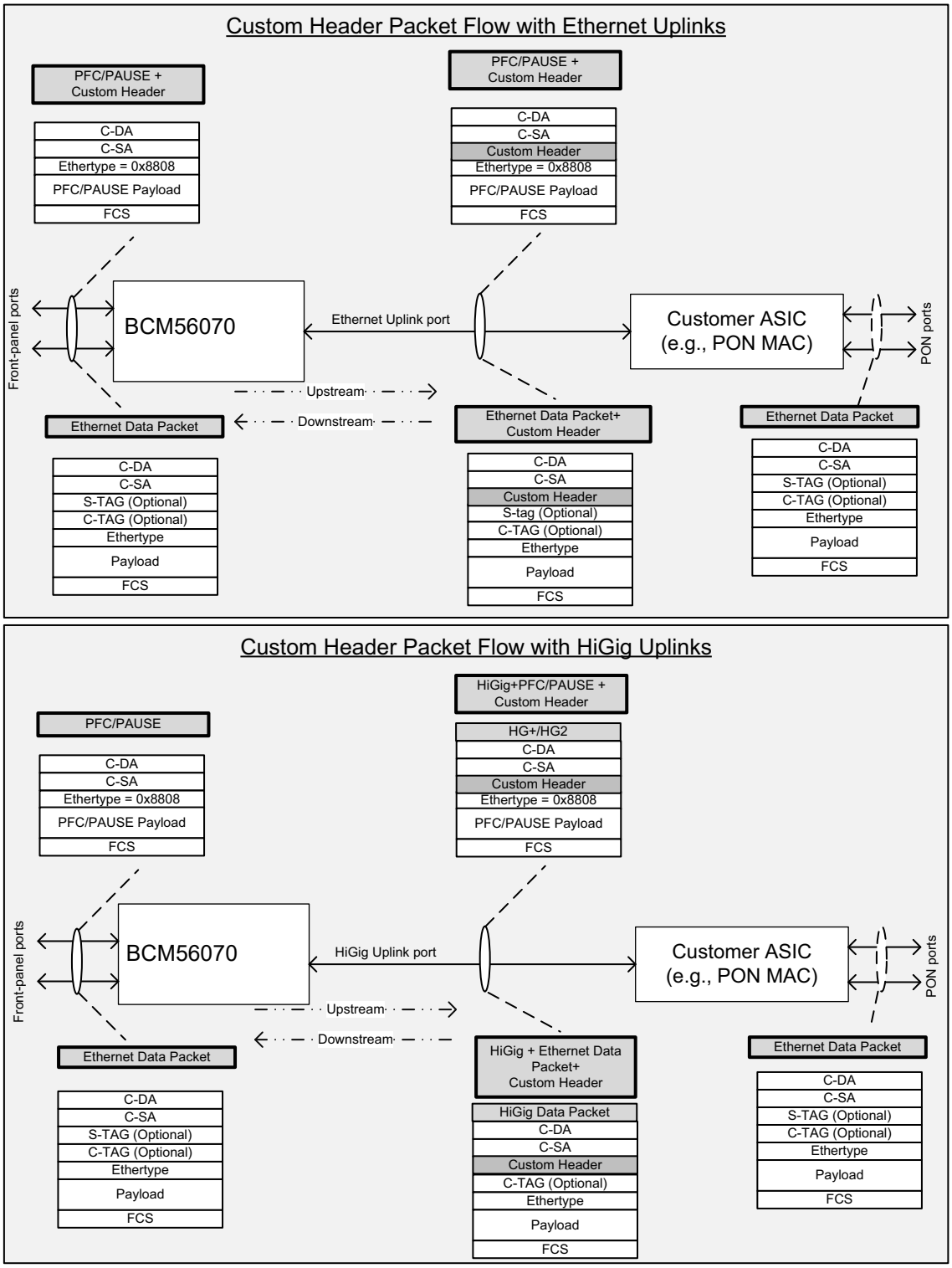
4.9.3 Custom Header Traffic Flows

Figure 31 illustrates the packet formats at various interfaces in systems supporting custom header with and without HiGig interfaces.

As illustrated in the upstream direction, the BCM56070 will not recognize custom headers in Ethernet packets received on downlink ports. In the BCM56070, of those Ethernet packets can be selected for custom header encapsulation through software configuration. Packets encapsulated with customer headers will be forwarded through uplink ports to a customer ASIC or CPU, potentially through an Ethernet network that is not shown in the diagram. If there is an Ethernet network between the BCM56070 and the customer ASIC or CPU, the custom header format content must be configured appropriately so that the packet can be transported through it. The custom header is used by the customer's ASIC or CPU either for control plane or data plane or flow control processing.

In the downstream direction, custom header may be recognized in packets received on an uplink port of the BCM56070 through configuration. If a custom header is found in a packet, it can be used in the BCM56070 to determine the destination port, or to influence QoS decision, or for triggering priority flow control on a downlink port. If the same packet is forwarded through configuration to a destination port, the custom header will always be removed by hardware from the packet. The behavior cannot be changed through configuration. If priority flow control is triggered by a packet with a custom header, the BCM56070 must be configured by software to drop the packet.

Figure 31: Custom Header Packet Flow



4.9.4 Custom Packet Header Upstream Processing

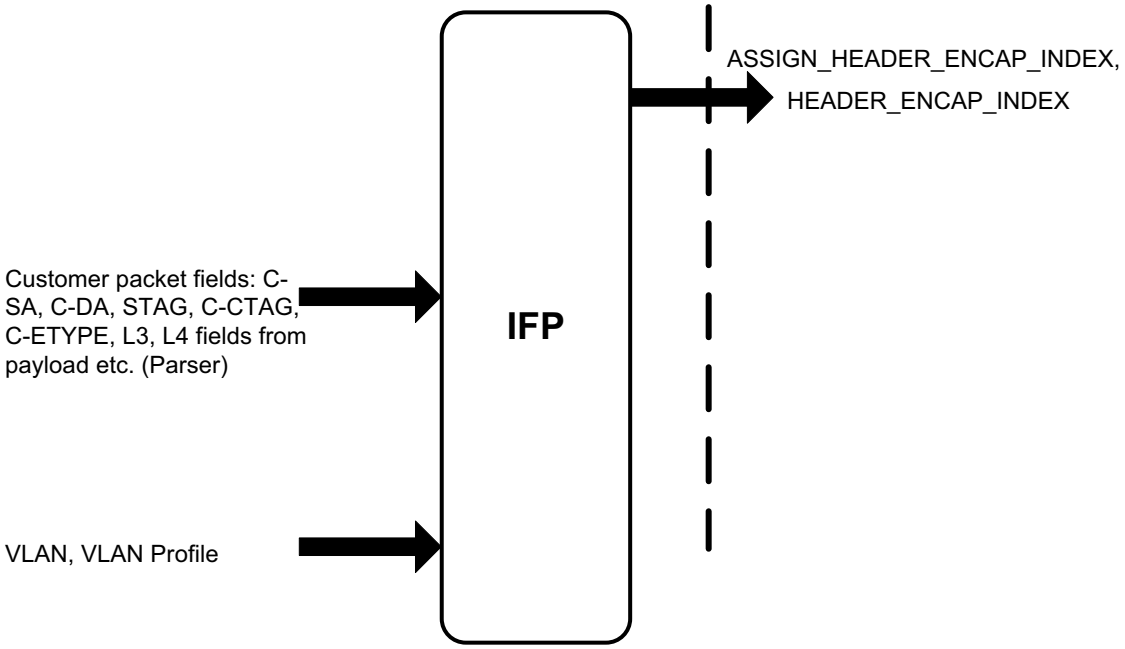
Figure 32 and Figure 33 illustrate how various functional blocks in BCM56070 process packets received by the BCM56070 in the upstream direction. In this direction, one or more of the BCM56070's downlink ports will receive Ethernet packets. The BCM56070's downlink port may receive Ethernet packets with or without HiGig header but not with custom header. In an application the CUSTOM_HEADER_ENABLE bit in the port table of each downlink port must be reset through software configuration. The only IPEP functions that have special functions related to custom headers are IFP and packet modifications. The following sub-sections describe those special functions in the BCM56070.

4.9.4.1 Upstream IFP Processing

Figure 32 illustrates upstream IFP processing of an Ethernet packet that may be modified to insert a custom header in the BCM56070 before being forwarded to uplink ports.

An IFP lookup of any upstream packet may result in an IFP action ASSIGN_HEADEDR_ENCAP_INDEX to add custom header to the packet. The data associated with the action is HEADER_ENCAP_INDEX that points to a table with fields in the custom packet header. The table is shared between MiML and custom header applications. The custom header obtained from the header encap table is inserted in an upstream packet after optional modification to insert a mapped version of internal priority by the Packet Modification function.

Figure 32: Custom Header Upstream IFP Functions



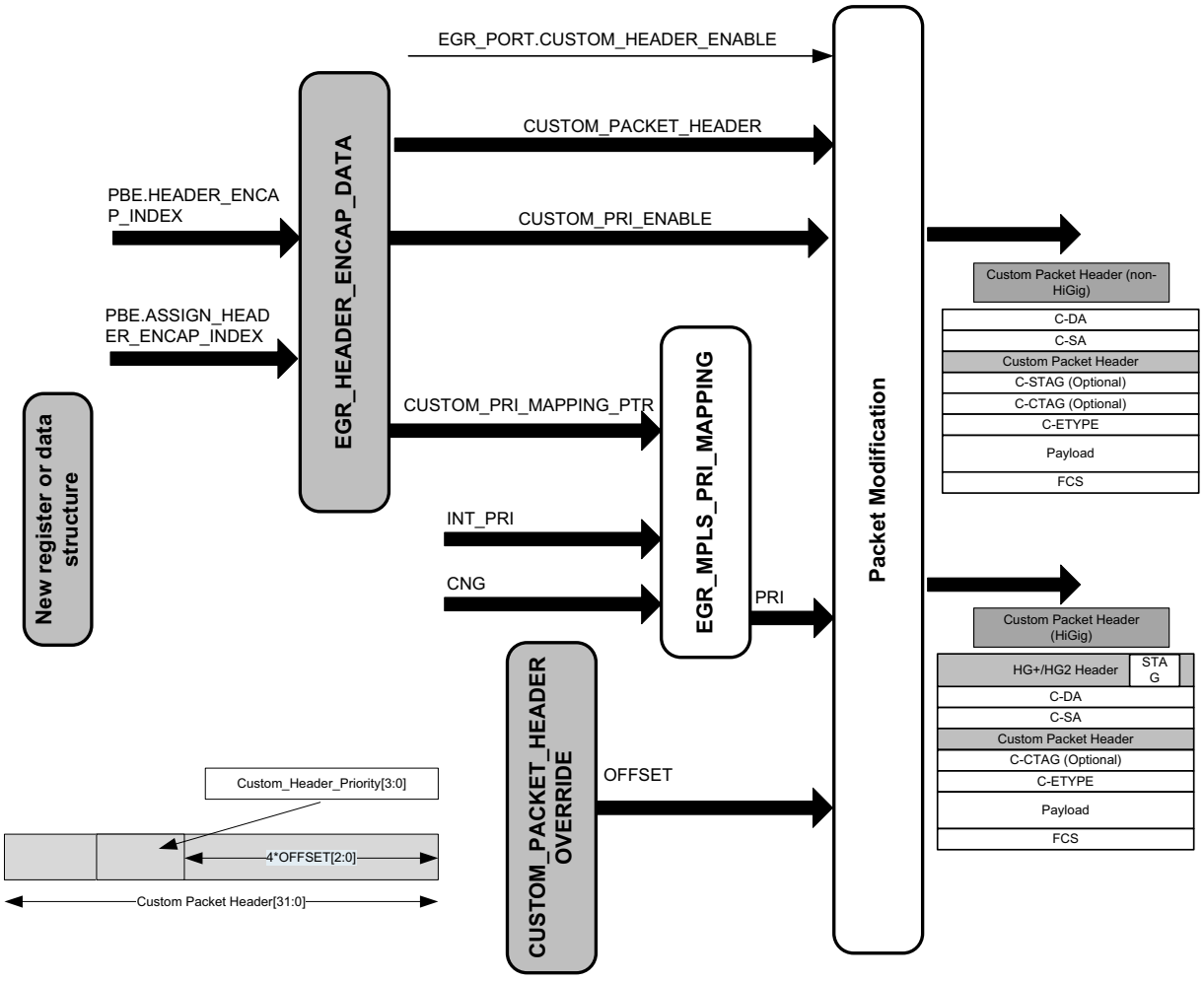
4.9.4.2 Upstream Packet Modification Function

A custom header is inserted in an upstream Ethernet packet if the ASSIGN_HEADER_ENCAP_INDEX IFP action is set for the packet and the CUSTOM_HEADER_ENABLE bit is set in the egress port table. Because of this, if an upstream packet is multicasted to multiple ports, then the IFP action ASSIGN_HEADER_ENCAP_INDEX may be optionally set for the packet but only the copies of the packet transmitted on egress ports that have CUSTOM_HEADER_ENABLE bit set will have custom header inserted by the Packet Modification function.

If a custom header is inserted in an upstream packet, other legacy packet modification functions like 802.1P, DSCP, remarking, and so on. may also be optionally performed by the Packet Modification function through legacy IPEP actions.

The associated HEADER_ENCAP_INDEX is used to obtain the custom packet header. Figure 33 illustrates how HEADER_ENCAP_INDEX is used to index the new, 128-entry EGR_HEADER_ENCAP_DATA table. The CUSTOM_PACKET_HEADER is the 32-bit custom header to be inserted in the packet after optional priority modification. Priority modification is opted by configuring the CUSTOM_PRI_ENABLE bit in the EGR_HEADER_ENCAP_DATA table. If the bit is set, the configurable CUSTOM_PACKET_HEADER_OVERRIDE register indicates a 4-bit aligned OFFSET in the packet where a 4-bit mapped internal priority is inserted in the 32-bit custom header. The mapped internal priority is derived by mapping internal priority and color in the EGR_MPLS_PRI_MAPPING table. The table has 8 mapping profiles and one of those is chosen for the packet by the lower 3 bits of the 4-bit wide CUSTOM_PRI_MAPPING_PTR from the EGR_HEADER_ENCAP_DATA table.

Figure 33: Custom Header Upstream Packet Modification Functions



4.10 Centralized Ethernet Switching (CES)

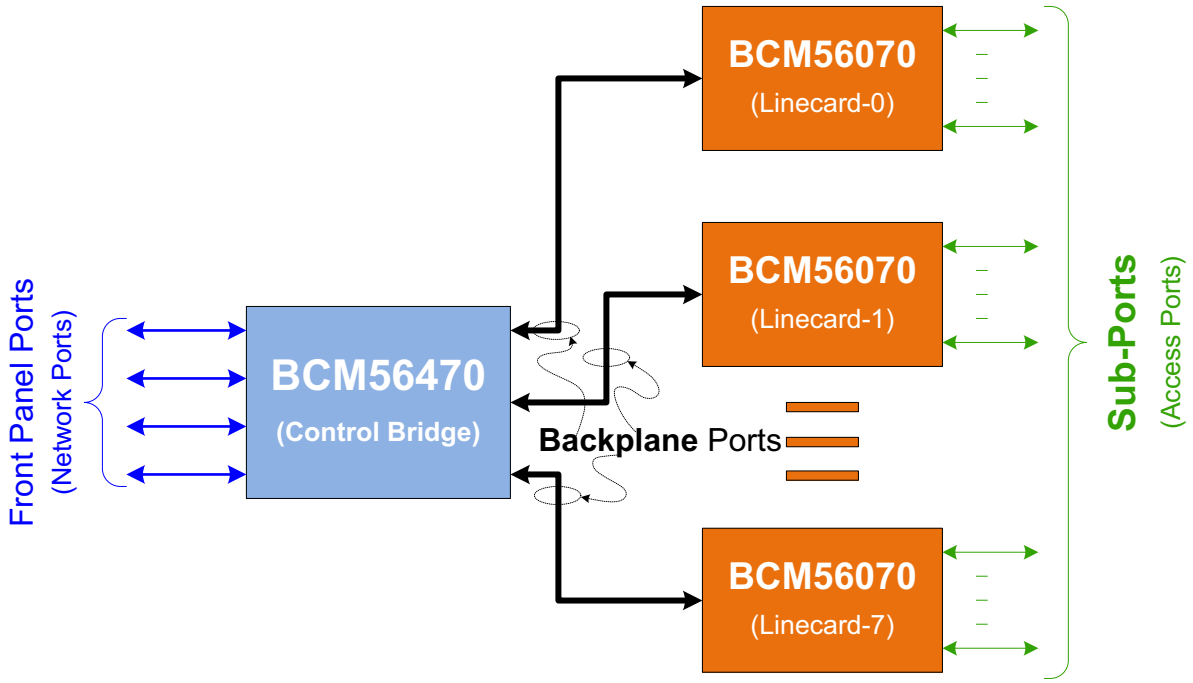
4.10.1 CES System Overview

One of the intended functions of BCM56070 is to operate as a line card in a Centralized Ethernet Switching (CES) system. This is shown in Figure 34.

CES systems are typically implemented in a chassis-based solution having line cards that serve as *port fan-out switches* and provide subtending ports (sub-ports or channel number) to expand the chassis' connectivity. In Figure 34, the BCM56070 devices can support three different sub-port configurations: 48 × 1G, 48 × 2G, or 12 × 10G. Assuming that the control bridge (CB) is a BCM56470 device, the system can be expanded with eight line cards resulting in a maximum of 384 sub-ports facing the Access domain.

In a CES system, the CB is responsible for the key functions for the entire system. It performs packet processing, makes forwarding decisions, handles replication for multicast packets, and takes care of queuing and traffic management for the entire system. The line cards, on the other hand, merely provide front-panel port for basic transport.

Figure 34: Centralized Ethernet Switching System



4.10.1.1 CES Backplane Interface

Each line card is connected to a CB through a high-bandwidth Ethernet interface called the *backplane port*. The backplane port must be configured to a higher bandwidth relative to the cumulative bandwidth of the active sub-ports within the line card. The reason for this requirement is to prevent congestion at the backplane interface both in the uplink direction and downlink direction. The system must be configured such that the backplane interface for both the CB and line cards never generate flow control (FC) signals due to the ingress buffer threshold being reached. This does not imply, however, that no

FC frames will ever be sent to the CB from the line cards. When a sub-port receives FC frames from its link partner, the line card redirects the FC message to the CB (which is expected to handle queuing and traffic management for the sub-port), and a copy is delivered to the local MMU to stop the proper TX queue. Any flow control frame that is sent to the CB should originate from the link partner of the sub-ports, rather than from the backplane interface of the line cards.

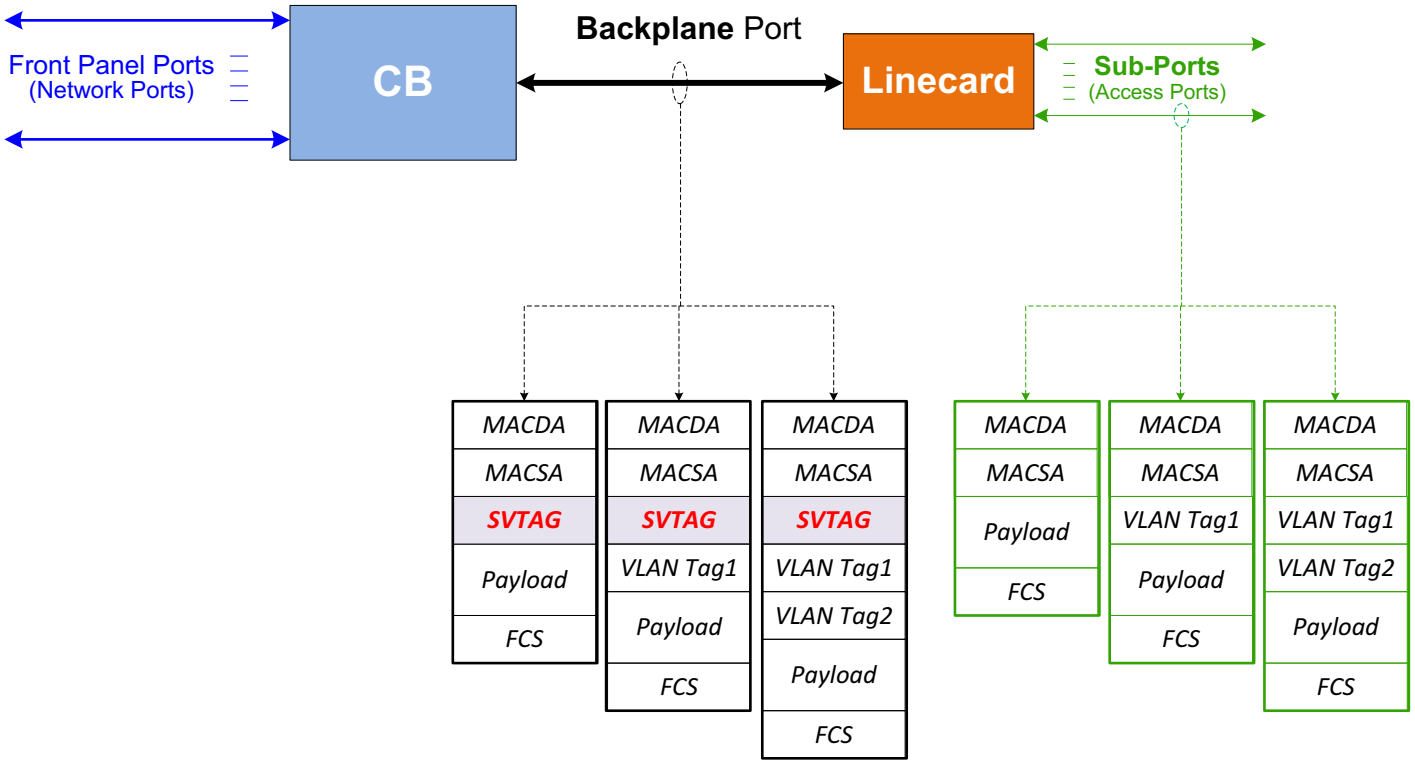
NOTE: Backplane ports on a line card cannot be a LAG because each sub-port maps statically to one of the backplane ports.

In a centralized chassis solution, the backplane ports are never identified as source ports or destination ports of any forwarding decision made by the CB. The backplane ports are treated as internal data bus and are transparent to the user. The goal is to make the chassis system behave like a single switch.

4.10.1.2 SVTAG

Packets within the backplane interface always include a 4-Byte Security VLAN Tag (SVTAG). As shown in Figure 35, the SVTAG is placed after the MACSA and before the first standard VLAN Tag if the original packet is single or double VLAN tagged. Depending on the packet's direction, the SVTAG is referred to as either *Source SVTAG* or *Destination SVTAG*. This distinction controls the content and purpose of the SVTAG. The term *source* or *destination* is taken from the perspective of the controlling bridge.

Figure 35: CES Packet Format



If the packet enters the backplane port of the CB, then its SVTAG (added by the line card) is called *source SVTAG*. This implies that the SVTAG contains information about the source sub-port or the port on which the packet entered the system.

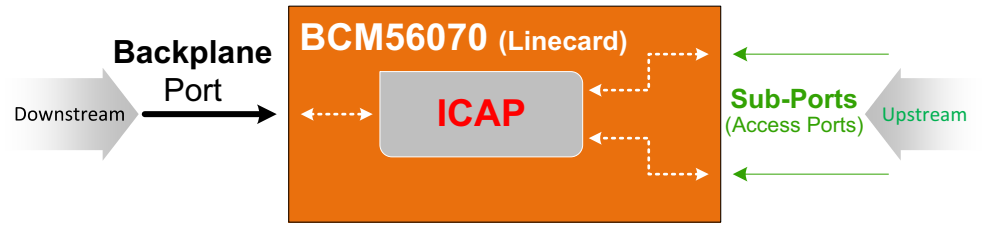
If the packet exits the backplane port of the CB, then its SVTAG (added by the CB) is referred to as *destination SVTAG*. This contains information about the destination sub-port or the port from which the packet exits the system.At the egress sub-port, the destination SVTAG is removed before the packet is sent to the wire.

4.10.2 CES Line Card

4.10.2.1 Line Card Packet Forwarding

In line card mode, the BCM56070 is expected to have very little functionality. It does not have the intelligence to do L2 and L3 lookups and forwarding decisions, and it has no traffic management capabilities. Packets coming in or out of the line card are forwarded to the egress port using the ICAP feature of the device. This is shown in the following figure.

Figure 36: Packet Forwarding in Line Card Mode



In the downlink direction, the destination sub-port is extracted from destination SVTAG and used as part of the FP qualifier. The FP action will then redirect the packet to the appropriate egress sub-port.

In the uplink direction, the ingress sub-port may be used as an FP qualifier to determine the appropriate egress backplane port, where the Source SVTAG is added to the outgoing packet. The FP will also indirectly provide the index to the HW resource from which the header properties of the outgoing SVTAG is derived.

4.11 Wireless Traffic Visibility and ACL Offload

4.11.1 Overview

A typical network consists of a combination of wired and wireless ports. If the BCM56070 is in an edge switch in such a network, it will be connected to one or more wireless Access Points (AP) as well as wired devices like a PC. The wireless traffic can be either Ethernet packets if the APs are standalone (they do all network processing) or could be encapsulated in a CAPWAP tunnel if a wireless controller exists in the network. Traffic from a standalone AP consists of Ethernet packets and the BCM56070 will process those packets the same way as it processes Ethernet packets from a wired port.

The new wireless client traffic visibility and ACL offload feature in the BCM56070 will support the following new optional features:

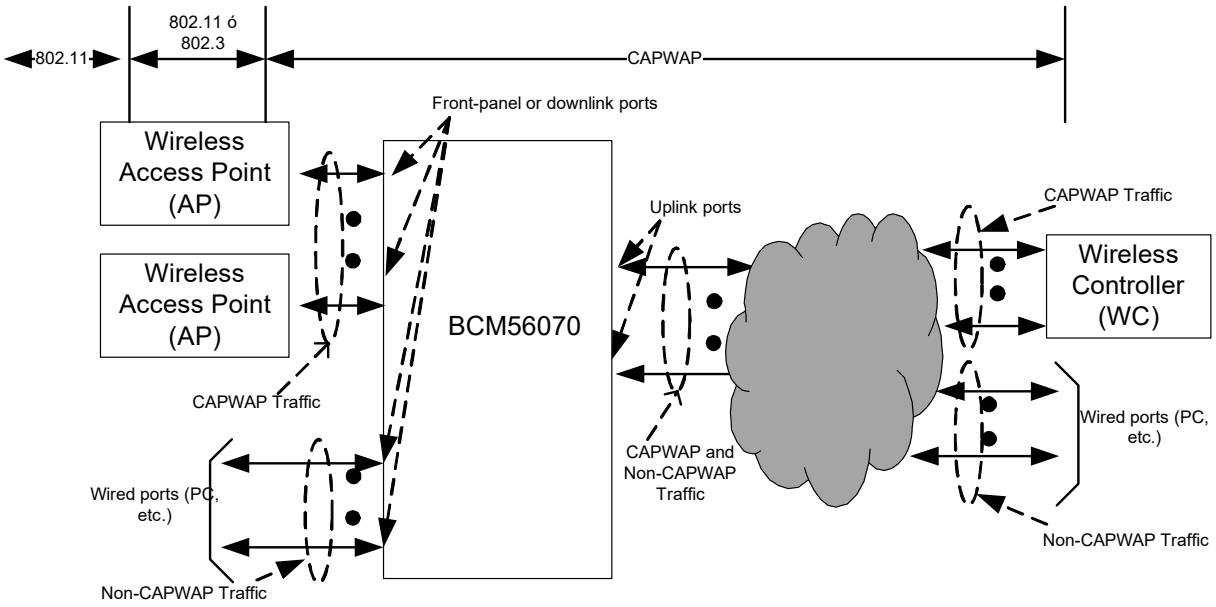
- Parse the wireless client packets inside the CAPWAP tunnel.
- Monitor wireless client traffic load using the new VLAN counters as Ingress VLAN translation table lookup action or the legacy IFP counters.
- Take one or more IFP actions based on wireless client packet headers.

No egress parsing or lookup function in the BCM56070 will support wireless client packet headers. In addition, the BCM56070 will continue to support all switch ingress as well as egress functions including the ones mentioned above based on the outer IP+UDP packet header of a CAPWAP packet.

A Wireless Controller (WC) terminates CAPWAP tunnels and processes the inner payload in the tunnel. A WC can be either centralized (see Figure 37) where it is shared across many edge switches in the network or it could be integrated in an edge switch. The BCM56070 does not terminate CAPWAP traffic - hence rest of the document will be for a centralized WC network.

As illustrated in Figure 37, a BCM56070's downlink port may either receive CAPWAP packets from AP or a wired port.

Figure 37: Centralized Wireless Traffic Processing



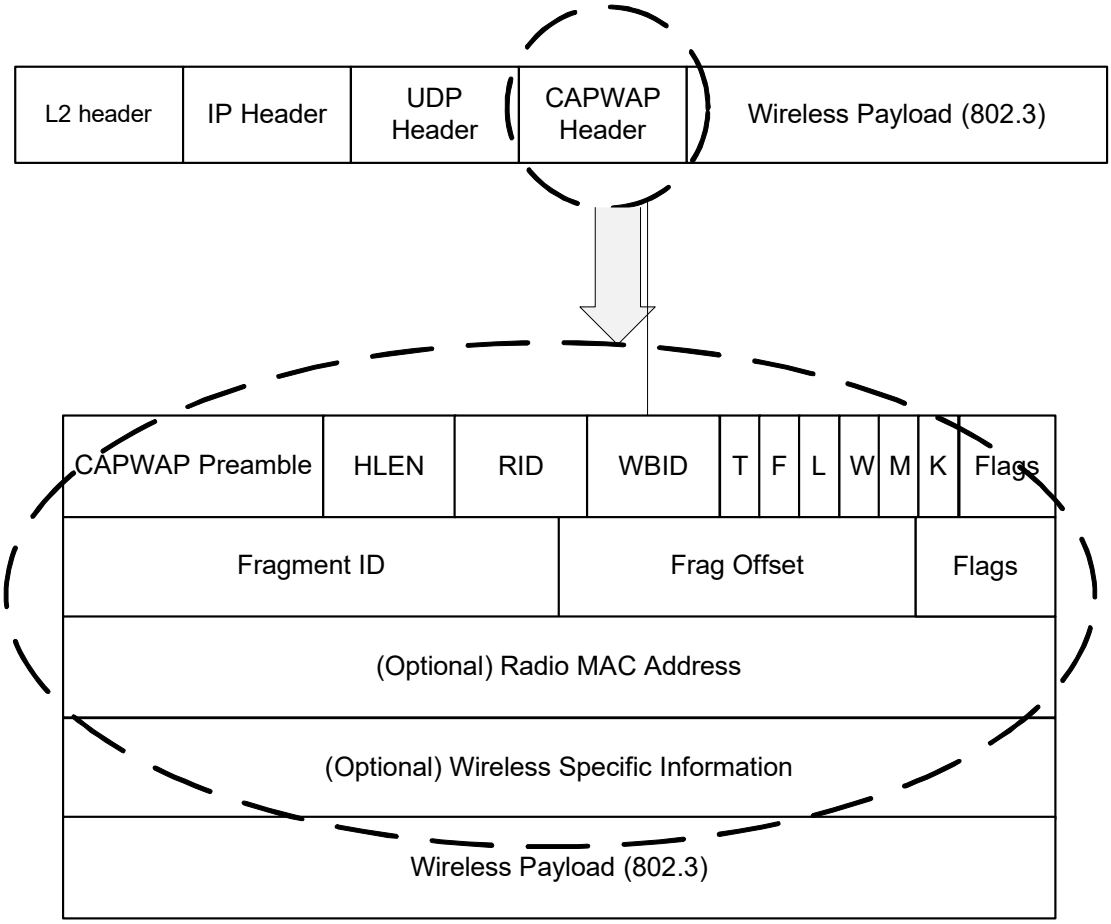
4.11.2 CAPWAP Packet Format

RFC5415 specifies the CAPWAP protocol. It defines a control protocol to setup one or more data tunnels between an Access Point and a Wireless Controller. CAPWAP control packets have the IPv4 or IPv6 packet format and UDP transport protocol. It uses well-known UDP destination port number 5246 in the outer tunnel header. The BCM56070 treats a CAPWAP control packets as any other IP or UDP packet.

A CAPWAP data tunnel can carry different types of client packets in it from various different types of wireless media. However, in the SMB and Enterprise edge deployments targeted with this new CAPWAP capability in the BCM56070, a CAPWAP data tunnel will wireless client data in native Ethernet format. A CAPWAP data packet has IPv4 or IPv6 header and uses UDP transport protocol. It uses well-known UDP destination port number 5247 in the outer tunnel header. By default, UDP-Lite transport protocol is used when IPv6 CAPWAP data packets are run over an IPv6 network. Refer to RFC5415 for more details.

A CAPWAP packet header is encapsulated in the IP or UDP tunnel along with the wireless payload in native Ethernet format. Figure 38 illustrates the CAPWAP packet header format.

Figure 38: CAPWAP Packet Header Format



The section below provides descriptions of the CAPWAP header fields and the set of values that BCM56070's CAPWAP header parser hardware supports:

- CAPWAP Preamble – The CAPWAP Preamble has a 4-bit version which is specified as 0 in RFC 5415 and a 4-bit type field. The 4-bit type field is 0 for CAPWAP header and 1 for DTLS header. Packets with CAPWAP header version 0 is recognized by the parser.
- HLEN – A 5-bit field containing the length of the CAPWAP transport header in 4-byte words (similar to IP header length). This length includes the optional headers. The parser parses the first two 4-byte words in the CAPWAP header and can skip additional two or four 4-byte words in the CAPWAP header before parsing the 802.3 payload. If the HLEN is > 6, the parser will not parse beyond the CAPWAP transport header and extract any Wireless Payload field.
- RID – A 5-bit field that contains the Radio ID number for this packet, whose value is between one (1) and 31.
- WBID – A 5-bit field that is the wireless binding identifier. The identifier will indicate the type of wireless packet associated with the radio. The value of 1 is for IEEE802.11 wireless packets and is relevant for CAPWAP based wireless traffic passing through the switch.
- T – The Type *T* bit indicates the format of the frame being transported in the wireless payload. When this bit is set to one (1), the payload has the native frame format indicated by the WBID field. When this bit is zero (0), the payload is an IEEE 802.3 frame. The parser can only parse 802.3 type payload (T = 0).

- F – The Fragment *F* bit indicates whether this packet is a fragment. When this bit is one (1), the packet is a fragment and MUST be combined with the other corresponding fragments to reassemble the complete information exchanged between the WTP and AC. The 802.3 payload in the first fragment of a CAPWAP packet can only be parsed by the hardware parser. Hence, features like the VLAN counters or IFP lookup that are based on the Wireless Payload fields are not available for non-first fragments of CAPWAP packets.
- W – The Wireless *W* bit is used to specify whether the optional Wireless Specific Information field is present in the header. A value of one (1) is used to represent the fact that the optional header is present. The parser ignores the value of the W bit and does not parse the optional Wireless Specific Information field is present in the header.
- M – The Radio MAC *M* bit is used to indicate that the Radio MAC Address optional header is present. This is used to communicate the MAC address of the receiving radio. The parser ignored the value of the M bit and does not parse optional Radio MAC Address field is present in the header.

The BCM56070 only supports parsing of Ethernet wireless payload. Since the BCM56070 can only parse at most the first 128 bytes of any packet, the types of SNAP or Ethernet II packets it can parse are limited to that parsing depth. The following are some of the additional limitations of wireless payload parsing:

- Either Ethernet II or SNAP packets in the wireless payload are parsed by the BCM56070's parser.
- Wireless Payload fields beyond 0, 1, or 2 VLAN tags are parsed.
- If the CAPWAP wireless payload is an IPv4 packet, the BCM56070's parser can parse over an Authentication Header.
- If the CAPWAP wireless payload is an IPv6 packet, the BCM56070's parser can parse over one extension header at most.

4.11.3 CAPWAP Traffic Flows

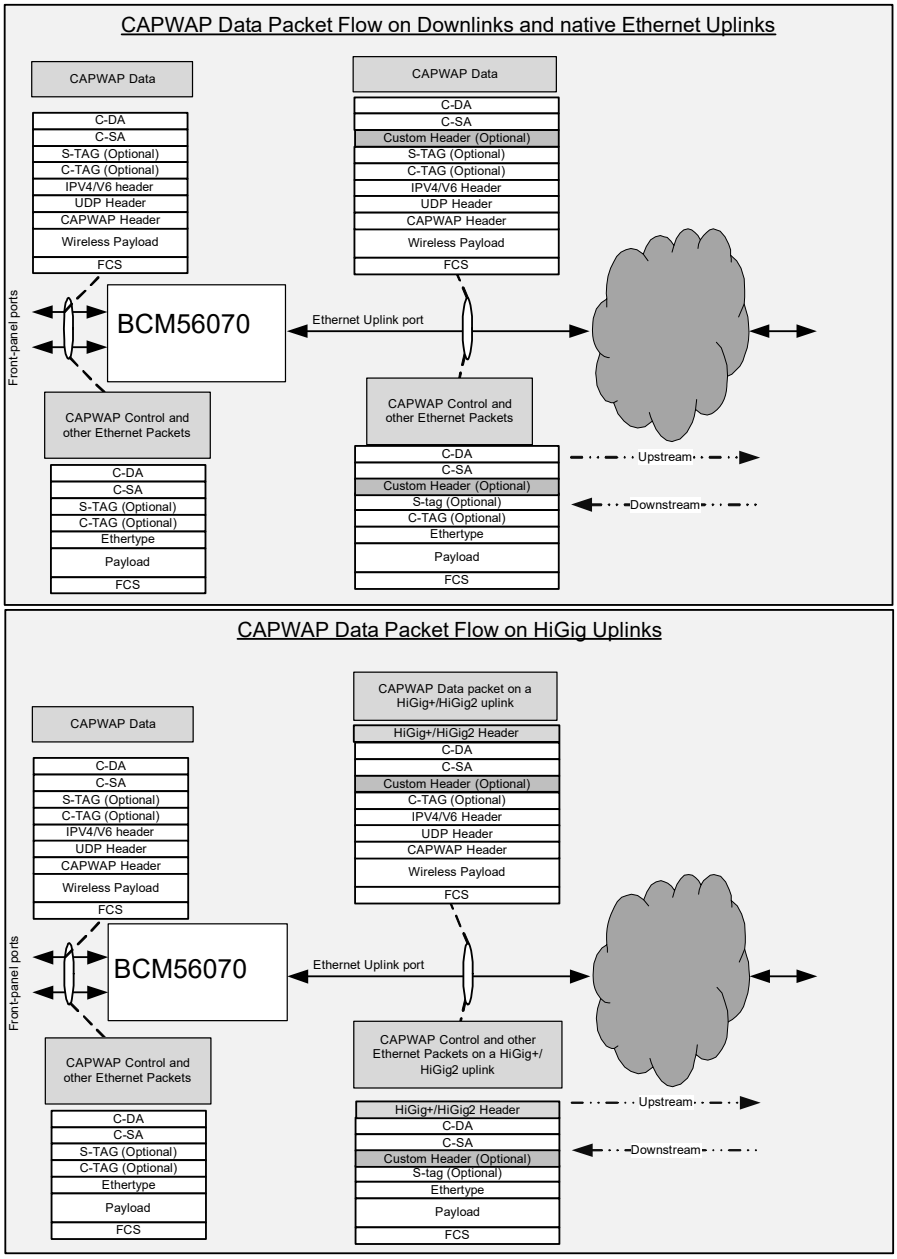
Figure 39 illustrates packet formats at various interfaces in systems wireless deployments supporting CAPWAP Wireless Controller architecture with Ethernet wireless payloads in CAPWAP data packets.

As illustrated in the upstream direction, the BCM56070 will recognize CAPWAP data packets with Ethernet II wireless payloads. It will only support optional IVXLATE and IFP actions based on wireless payload packet fields. After ingress processing, a custom header may be optionally added to the packet in the egress path of the BCM56070 before it is transmitted out on either a native Ethernet or a HiGig uplink. Although the BCM56070 has the capability to encapsulate any packet with a Mac-in-Mac (MiML) packet header, there is currently no use case for the BCM56070 hardware to parse and act on CAPWAP and wireless packet header fields and encapsulate the same packet in MiML.

In the downstream direction, the BCM56070 will recognize CAPWAP data packets with Ethernet II wireless payloads received on a native Ethernet or a HiGig uplink. It will only support optional IVXLATE and IFP actions based on wireless payload packet fields for native Ethernet packets. However, if a downstream packet is received on a HiGig uplink then only IFP actions are available because IVXLATE lookups are not supported for wireless payload in a CAPWAP packet transported over a HiGig link. After ingress processing, the custom header in the packet will be de-capsulated if it is present and then will be transmitted out on one or more native Ethernet downlinks as a CAPAWAP data packet.

LAG resolutions of CAPWAP packets are done using fields from the outer header of a CAPWAP packet. In neither upstream nor downstream direction the wireless payload's header fields will be used for lag resolution.

Figure 39: CAPWAP Packet Flow



Chapter 5: Load-Balancing

5.1 Stacking and Related Information

The device supports XFI-based uplink ports. These uplink ports are capable of operating at 1G, 2.5G, 5G, 10G, or 13G. The 1G and 2.5G configuration will operate on a single XFI lane(0) with HiGig-Lite™ support at these speeds. HiGig2 is supported at speeds of 13G.

The HiGig2 stacking protocol is Broadcom's proprietary interconnect scheme. The protocols use a standard XFI interface (just as 10GbE), with four RX differential lanes and four TX differential lanes.

In HiGig2 mode, the port does not run standard 10GbE and can only be used to interconnect other StrataXGS® II, StrataXGS® III, or StrataXGS® IV devices. HiGig2 is the next generation of the protocol which allows for extending a greater variety of features across multiple modules. These features include: mirroring, trunking, VLAN membership, and congestion awareness. The stacking support is a critical feature for seamlessly supporting a large number of metro users. Unless connecting with legacy devices is required, the use of HiGig2 is recommended. However, within a stack, HiGig and HiGig2 can be mixed if HiGig2 devices are restricted to the first 64 module IDs.

The HiGig2 protocol defines a module header that is used by the devices to properly forward packets across modules and communicate other information required to extend the StrataXGS features across multiple devices. Up to 64 modules can be interconnected, not including fabric devices (BCM56700, BCM56720, BCM5675, and BCM5676). Any number of fabric devices can be used, as they are not assigned a module ID.

This header is at the front of the Ethernet payload and contains the module fields. The VLAN tag (4 bytes) is extracted from the packet and the VLAN information (VID, CFI, and PRIORITY) is inserted into specific fields in the module header. The normal packet FCS field (32-bit CRC) is regenerated to cover the module header.

In scenarios where more than one HiGig2 port is used to connect to another StrataXGS IV device, HiGig trunking can be used to distribute traffic across the connected links. For unicast packets destined for a remote module, the MODPORT_MAP table is used to determine the appropriate HiGig port. For L2MC and IPMC, the HiGig port is included as part of the L2 bitmap in their respective multicast tables. For Broadcast or DLF, the HiGig port is included in the VLAN membership. Broadcast and multicast packets are also subject to the non-unicast Block Mask prior to HiGig trunk resolution. In addition, these tables also carry a HiGig Override bit in the table entry.

After the HiGig port bitmap is determined, a comparison is performed to see if any of the HiGig ports are a member of a trunk group (defined by HiGig_Trunk_Group and HiGig_Trunk_Control registers). If so, the HiGig trunk logic uses the RTAG value to determine which HiGig port to use. The HiGig port bitmap is rewritten. If the HiGig Override bit is set, no trunking is performed and the packet is sent according to the original HiGig port selection.

5.2 Link Aggregation (Trunking)

The following sections provide information about link aggregation (trunking).

5.2.1 IEEE 802.3AD Link Aggregation

Link aggregation or trunking is a mechanism by which several ports can be bundled together to form a single logical link or a trunk. This is useful when higher bandwidth or redundancy (or both) between switches are required. Link aggregation (LAG) includes support for the following:

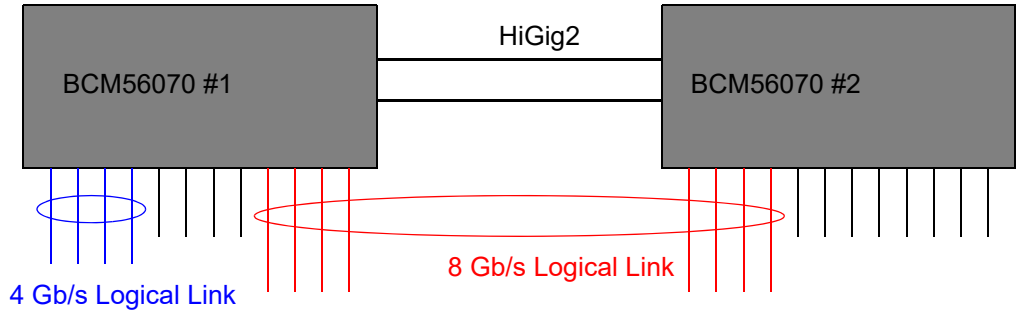
- Trunk port members are always configured for full-duplex.
- Trunking of GbE ports provide aggregate throughput up to a maximum of 16 Gb/s (eight member ports), full-duplex.
- Provides incremental bandwidth depending upon the system requirements (2 Gb/s increments).
- Provides link redundancy. In case of trunk port failure, the hardware can be preconfigured to automatically failover to a different set of ports, based on PORT_LAG_FAILOVER_SET table.
- Provides traffic distribution across trunk members.

The BCM56070 supports 128 trunk groups, with each group having up to eight member ports. The trunk resolution is determined using a hashing function based on a programmable combination of packet fields: MAC DA, MAC SA, VLAN, EtherType, IP DA, and IP SA.

The device supports Link Aggregation with no adjacency limitation within same switch module and also across HiGig2 links.

The BCM56070 device supports IEEE 802.1ad link aggregation. Up to eight ports can be assigned in a trunk group and up to 128-trunk groups are supported. In a large system, there can be up to 64 devices. Each device is assigned its own unique module ID, thus allowing link aggregation to span multiple devices.

Figure 40: 4-Port and 8-Port Trunk Group Example



5.3 RTAG7 Hashing

RTAG7 is a hashing algorithm used by the BCM56070 to load-balance traffic within a trunk group in a controlled manner. The hashing operation is based on load-balancing in split micro-flows that balances the traffic across multiple output links. Hashing-based load balancing in BCM56070 devices is used in the following applications:

- L3 ECMP (equal cost multipath)
- LAGs
  - Link Selection
  - Failover Link Selection
- HiGig trunking
  - Link Selection
  - Failover Link Selection

The RTAG7 Hashing is required for BCM56070 to support the following features:

- VXLAN (ECMP)
- RoCE v1 (LAG)
- RoCE v2 (ECMP)

Supporting VXLAN ECMP and RoCE (v1 and v2) LAG or ECMP implies that BCM56070 supports hashing for basic L2 and L3 flows.

5.3.1 RTAG7 Overview

The RTAG7 hash scheme generates a hash that consists of the following two portions:

- The first portion is generated primarily from packet headers to identify micro-flows in the traffic.
- The second portion comes from static physical configuration such as source and destination port numbers.

Each application can choose either portion based on user configurations.

The hashing operation is divided into the following three steps:

1. Hash Field Selection: Based on EtherType and packet parsing.
2. Hashing Computation: Based on configuration.
3. Hash Bits Selection: Individual application-based controls to select the hash.

Chapter 6: Network Management

6.1 Time-Sensitive Networking

6.1.1 TSN Overview

Time-Sensitive Networking (TSN) deals with networking of end points which are sensitive to or aware of time. The following table captures the IEEE standards being specified or enhanced in support of TSN and the functions and features required to achieve time synchronous networks.

NOTE: TSN supports a single-chip system configuration and does not support systems in multi-chip using HiGig stack and HiGig proxy.

Table 5: Standards Supporting TSN

IEEE Standards	Functions and Features
IEEE Std 802.3br-2016[a]	Interspersing Express Traffic
IEEE Std 802.1Qbu-2016[b]	Frame Preemption
IEEE Std 802.1Qbv-2015	Enhancements for Scheduled Traffic
IEEE Std 802.1Qca-2015	Path Control and Reservation
IEEE 802.1AS-Rev	Timing and Synchronization for Time Sensitive Applications - Revision
IEEE 802.1CB	Frame Replication and Elimination for Reliability
IEEE 802.1Qcc	Stream Reservation Protocol (SRP) Enhancements and Performance Improvements
IEEE 802.1Qch	Cyclic Queuing and Forwarding
IEEE 802.1Qci	Per Stream Filtering and Policing
IEEE 802.1CM	Time Sensitive Networking for Front haul
IEEE 802.1Qcp	YANG Data Model
IEEE 802.1Qcr[c]	Asynchronous Traffic Shaping

a. The device does not support IEEE 802.3br.

b. The device does not support IEEE 802.1Qbu.

c. The device does not support IEEE Qcr.

6.1.2 Introduction

IEEE 802.3 Ethernet is a packet-based statistically multiplexed medium without any awareness of absolute time for packet transport. Time sensitive traffic like voice and video are prioritized over other background, best effort traffic on Ethernet media to reduce end-to-end packet transport delay. TSN standards leverage circuit-switching concepts to introduce time awareness in Ethernet transport. The set of new TSN standards (Table 5) makes Ethernet suitable for time-sensitive traffic to mix with best-effort traffic in industrial Ethernet, wireless fronthaul, and vehicular networks that would otherwise require proprietary transport technologies like Ethercat, CPRI, and so on.

IEEE TSN standards guarantee time sensitive, time aware, and reliable communication among processing elements and end points on the same Ethernet medium that also carries background Internet traffic. Benefits and applications include the following:

- The IEEE 802.3BR-2016 and IEEE 802.1Qbu-2016 standards help reduce packet transport delay and jitter of time sensitive traffic by assigning a new *Expedite* class and permitting preemption of other traffic.
- IEEE 802.1Qbv-2015 introduces time aware scheduling of control traffic in industrial automation, process control, and vehicle control.
- The IEEE 802.1CB standard guarantees ultra low packet loss of loss-intolerant traffic flows through replication of all packets of the flow at the transmitter, transport of duplicates through different paths of the network, and duplicate discard at the receiver.
- IEEE 802.1Qci draft allows filtering traffic from babbling talkers to protect all other communication.
- IEEE 802.1AS-Rev draft creates a common time reference by synchronizing time in talkers and listeners in a TSN network.

Although this document describes some of the common TSN applications like Industrial Automation and Wireless Fronthaul transport, TSN technologies are also well suited for applications like process control, high-fidelity audio systems, autonomous driving, and so on. This document also describes hardware architecture of TSN technologies implemented in this device in compliance with IEEE TSN standards and drafts.

6.1.3 TSN Application Areas

6.1.3.1 Industrial Automation and Process Control

Real time traffic requires low latency transport over Ethernet. Various applications areas like synchronized axles and drives, power generation, transmission and distribution networks, and the transportation industry are a few examples where real-time traffic is important. Cycle times for the transmission of time-sensitive process data are often significantly below 1 millisecond. Various non-Ethernet technologies like EtherCAT, Profinet IRT, SERCOSIII, and so on. have evolved over time as low latency and low jitter transport technologies in Industrial Ethernet. Some of these technologies use non-standard mechanisms over conventional Ethernet to provide latency guarantees and in some cases standard Ethernet Switch, MAC, or PHY. These technologies are often incompatible with each other. Instead, a more unified Ethernet network will ride on Ethernet cost curve, and technology investments will lead to potential cost savings as well as investment security when opting for the implementation of real-time Ethernet. This is key for industrial applications such as process and machine control where low communication latency and minimal jitter are critical to meeting closed loop control requirements. TSN is the first fully open, standard and interoperable way to fulfill these requirements.

Manufacturing operations requires tight coordination of sensing and actuation to safely and efficiently perform closed loop control. Typically, these systems have been deployed using non-standard network infrastructure or air-gapped (unconnected) standard networks. This approach makes devices and data much harder to access and creates a technical barrier to Industrial Internet of Things (IIoT), which is predicated on the ability to consume data anywhere throughout the infrastructure.

Productivity and product quality determine market success. Precise motion control, dynamic drives, high-speed controllers, and the deterministic synchronization of devices are key factors in achieving superior production. They allow high production speeds and simultaneously optimized product quality. This is made possible by increased networking of the sensors and actuators that are involved in the production processes. Another factor is the increased integration of the (local) Cloud, where, for example, virtual programmable logic controllers are hosted and interact directly with the production process through the sensors and actuators at the field level.

TSN enables a single, open network infrastructure supporting multi-vendor interoperability through standardization and IT and OT convergence through guarantee of service. The technology will be used to support real-time control and synchronization of high-performance machines over a single, standard Ethernet network.

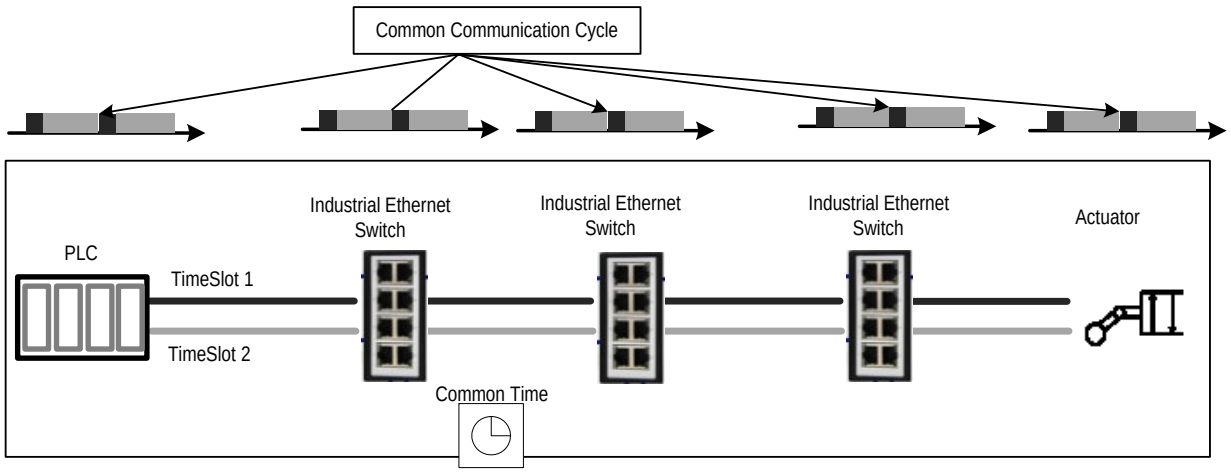
Industrial networks which constitute programmable logic controllers, sensors and actuators are typically connected as a chain or as a ring (for redundancy). The control communications on these networks are done at regular intervals called machine cycles. In order to enable remote management of these devices, the control and administrative communication are allowed to happen over the same network. The control traffic is scheduled at regular intervals from the controller, where the monitoring is done with the sensors and the response commands are sent to the actuators.

In typical industrial networks, the machine cycle time is 125 μs. Some of the systems like CAM's using a tighter machine time (62.5 μs). Half this cycle time is typically consumed for computations using the sensor data and determine the next set of commands to be issued to the actuators. This leaves about 32 μs for the communication path. With sub microsecond switch latency, achievable with this device, a single PLC controller can manage a network with up to 30 hops (for a 62.5 μs machine cycle). The time aware policing (IEEE 802.1Qci) and scheduling (IEEE 802.1Qbv) features allow for these communication channels to co-exist without impacting the control loop requirements. The interspersing of express traffic using preemption (IEEE 802.3br and IEEE 802.1Qbu) allow for minimizing the packet delay variation due to interfering frames. The bandwidth available outside the guaranteed machine cycle communication can be used for diagnostics and fault detection.

Quickly identifying the location of the fault allows for minimal downtime of these industrial networks. Orchestrating the up and down cycles of the industrial machinery can also result in efficient use of power. Figure 41 illustrates one such communication cycle where control and administrative traffic is mixed on the same link. Timeslot 1 corresponds to the periodic control traffic, while Timeslot 2 corresponds to background lower priority traffic (which could be carrying diagnostic or any other administrative information).

For any of the time-based orchestration requires all components in the networks requires a common reference time. Timing and Synchronization with IEEE 802.1AS-rev or IEEE 1588v2 Precision Time Protocol (PTP) allows for phase synchronizing an Ethernet network for TSN applications. Applications with high accuracy requirements should get the frequency reference either using SyncE or high precision clock sources for better synchronization performance.

Figure 41: Machine Cycle in Industrial Network

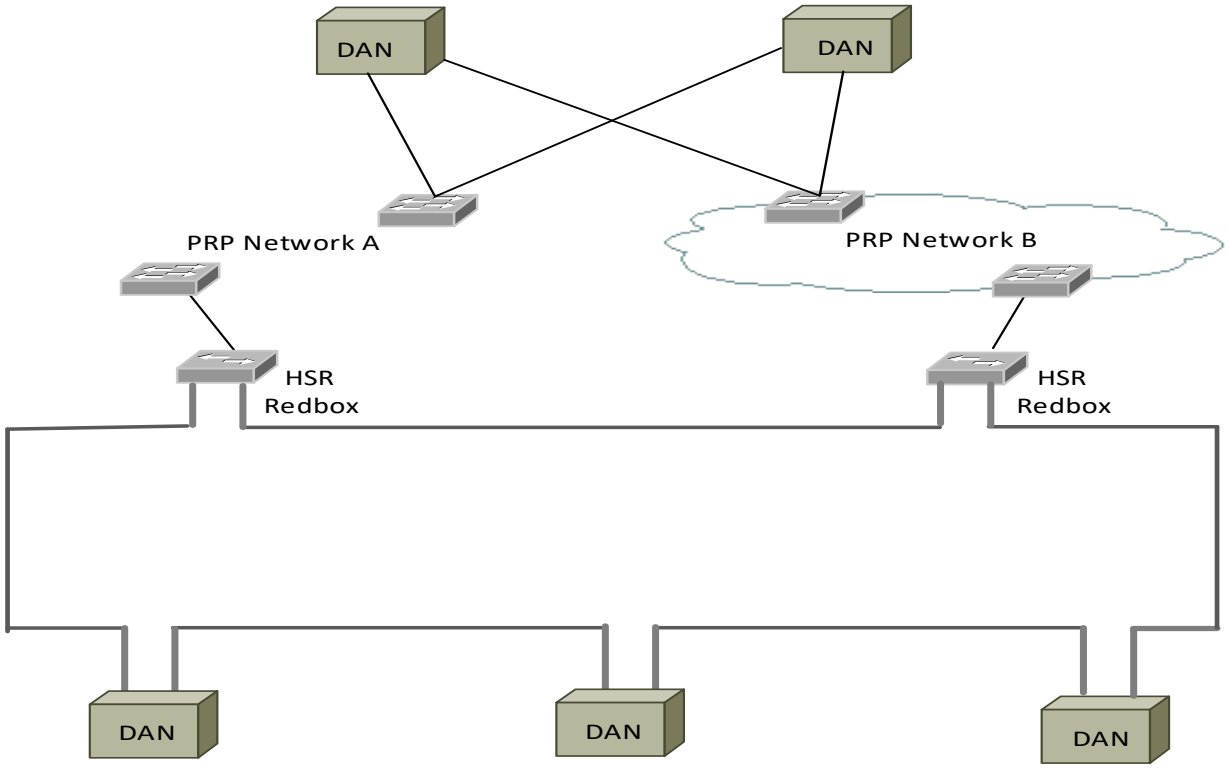


Deterministic data transmission is subject to the risk of failure of a network component or cable. There are well established redundancy mechanisms that are already being used in the industry to protect the network from the packet losses resulting from such an interruption. High Availability Seamless Redundancy (HSR) and the Parallel Redundancy Protocol (PRP) are some of the well established mechanisms. Figure 42 illustrates Dual Attached Nodes (DAN) connected in a network with redundancy mechanisms (both at network level with PRP across rings as well as within a ring with HSR).

The IEEE standard 802.1CB uses a similar mechanism that uses static redundancy procedures, where redundant transmission paths are permanently active. The advantages of static redundancy mechanisms avoid the switch over times that are present in dynamic redundancy procedures in the case of a failure. In comparison to HSR and PRP, the redundancy mechanisms developed in IEEE P802.1CB offer the advantage that they can be used in any topology. IEEE 802.1CB is neither limited to a ring topology or topologies with completely independent networks, nor restricted to exactly two redundant paths. In order to reduce the probability of a packet loss, it is also possible with IEEE 802.1CB to utilize numerous redundant transmission paths within the bounds of latency guaranties of the application.

This device allows for any of the above protocols to be deployed in an industrial network. This enables both backward compatibility with existing networks, and forward compatibility with TSN deployments.

Figure 42: Redundancy Networks with HSR or PRP

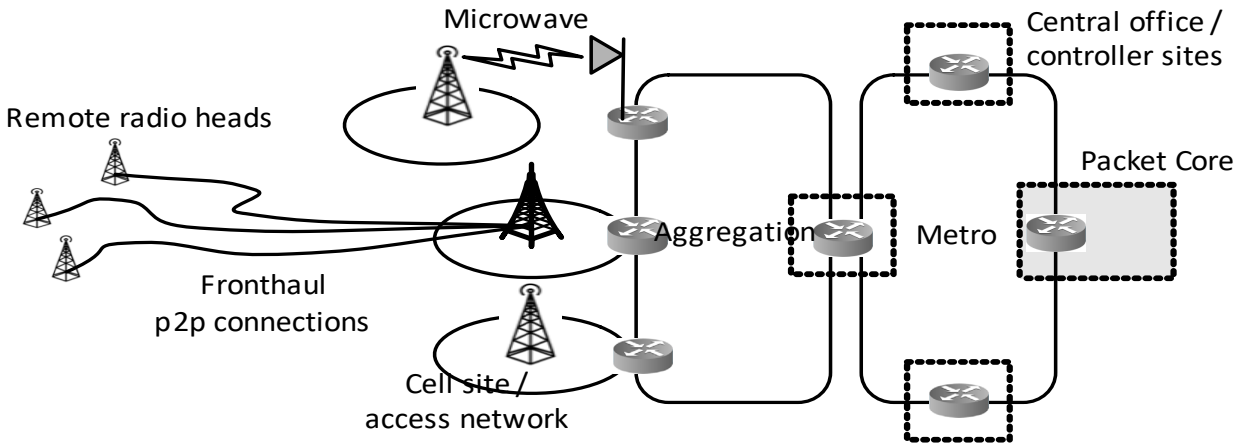


6.1.3.2 Ethernet-based Fronthaul and XHaul Networks

Mobile networks traditionally use proprietary, non-Ethernet technologies to connect baseband processors with radios. Those proprietary technologies will not support cost effective and scalable networks for next generation wireless technologies. Mobile operators are looking at new solutions that can help to simplify networks, boost the radio access performance and reduce cost, by sharing resources, using mainstream transport technologies, applying modern data center design where applicable and enabling coordinated radio technologies. These led, for example, to centralized radio access network

(C-RAN) deployment models and the separation of remote radio heads (RRH) and the radio base band processing unit (BBU). The high bandwidth and latency sensitive transport link between the BBU and RRHs is referred to as Fronthaul (FH). The latencies are hard bound by the allocated time budget for the radio processing and how much the BBU consumes it. The excess time can be used as the latency budget for the fronthaul. The following figure illustrates this for the 4G radio access networks.

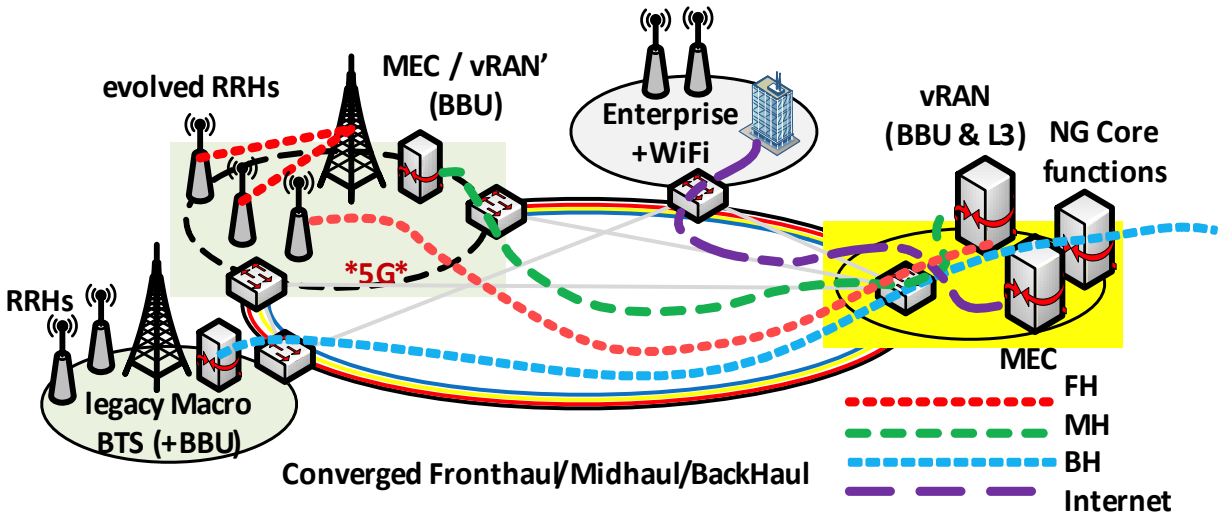
Figure 43: Legacy Mobile Network Topology



For implementing Fronthaul over packet networks such as IEEE 802.1 bridged networks, it is essential to meet the stringent latency bound and time-synchronized service requirements of protocols running over the Fronthaul. However, in the future the same transport network infrastructure should also service Internet like Backhaul (BH) traffic. A legacy deployment model with separate networks for Fronthaul and Backhaul as shown in Figure 43 is not feasible anymore. In addition, next generation 5G wireless networks will rely heavily on co-ordination of radio and baseband processing resources to improve quality of service. Midhaul refers to communication among those resources.

XHaul is a converged Ethernet transport network that will carry Fronthaul, Midhaul, and Backhaul traffic in a mobile wireless network assuring quality of service at the UE. The XHaul network premise is to have an end-to-end unified networking technology (for example, Ethernet) and a unified OAM throughout the entire RAN transport network. This would be an ideal platform to deploy other traffic engineering solutions on top of, including anything that can be done by applying the envisioned 5G network slicing paradigm in the context of the 4.5G or 5G RAN transport network. The challenge is that different use cases (for example, Backhaul versus Fronthaul versus Internet traffic) may have significantly diverse quality of service (QoS) and service-level agreement (SLA) requirements, which typically is challenging to the networking hardware. Furthermore, the RAN transport network may not be under a single administration, and even the ownership may be spread among multiple entities. The varied use cases and administration challenges mean that the XHaul network equipment faces stringent resource isolation and platform virtualization requirements.

Figure 44: XHaul Network Topology



The specification of the use of TSN features in XHaul scenarios is beneficial for vendors offering or developing TSN products, as well as for mobile operators. Within the 5G RAN switch, latency is just one component. The more pivotal features relate to network synchronization, deterministic latency, and reliability.

When Frame Loss Ratio (FLR) on the XHaul has to be minimal, IEEE 802.1CB Frame Replication and Elimination for Reliability (FRER) could be used. Potential use cases for the FRER include the 5G Ultra Reliable Low Latency Communication (URLLC) deployments. The FRER principle is discussed further in subclause 3.2. The usefulness of the FRER depends on the use case and the deployment. Due to potential FRER caused packet replication over alternative paths it may significantly increase the bandwidth requirements.

In summary, the IEEE 802.1TSN toolbox offers a wide range of usable tools, specifically to minimize FDV and give guarantees that the high priority (digitized radio) traffic gets through a switch in due time. These are important measures to meet in XHaul latency budgets, specifically if a network edge has to do *de-jitter buffering* (which shows as additional latency).

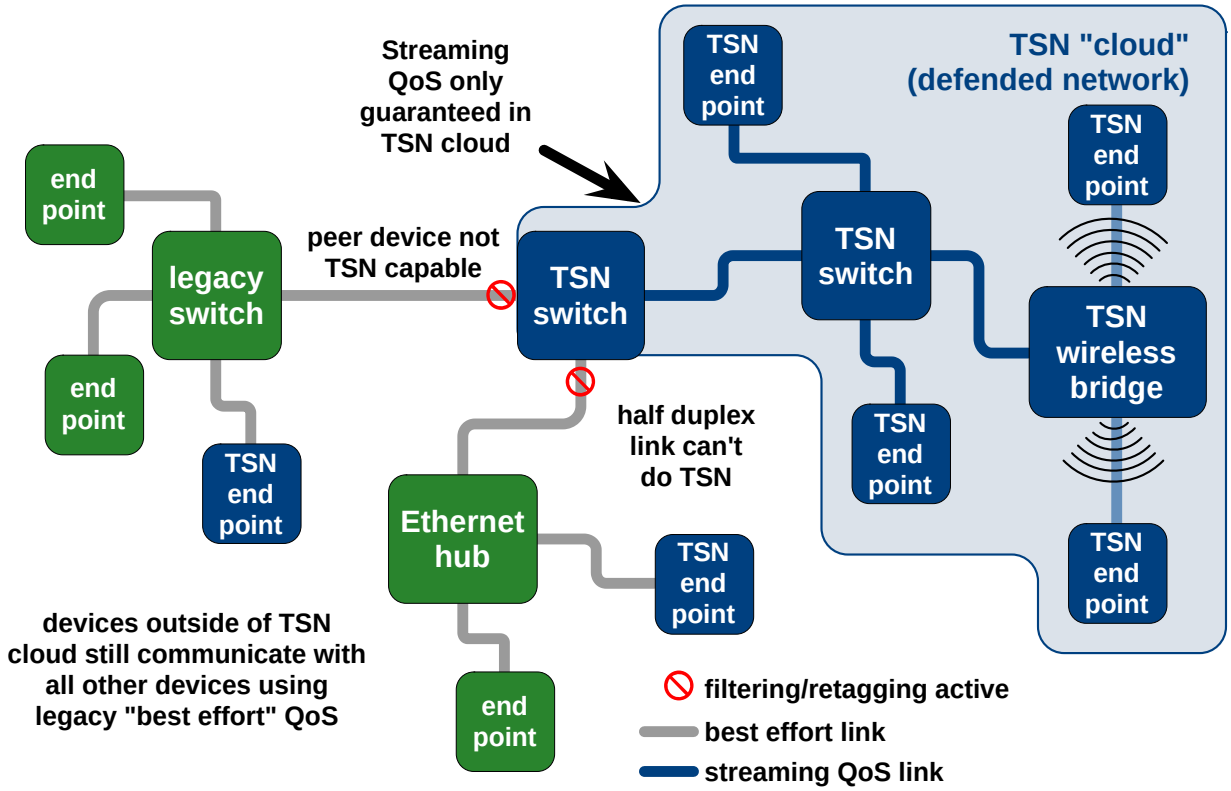
IEEE 802.1CM network profile for bridged Fronthaul networks has documented synchronization requirements for various categories of radio technologies: Category A+ (12.5ns - e_{ts}), A (45ns - e_{ts}), B (110ns - e_{ts}) and C (1.48us - e_{ts}), where e_{ts} is the timestamping error of the radio equipment (RRH). Accurate timestamping in the context of network synchronization has become one of the key features in XHaul targeted bridges.

6.1.4 TSN Hardware Features

6.1.4.1 Timing and Synchronization for Time-Sensitive Applications

Network synchronization is an integral part of TSN and time-sensitive applications. Therefore, the presence of relevant network synchronization protocols, such as IEEE P802.1AS or IEEE 1588 (version 2), can be used to achieve phase synchronization. For a frequency reference, the presence of ITU-T Synchronous Ethernet (SyncE) is likely. The accuracy of the network synchronization distribution has an extremely important role, specifically in time distribution networks with multiple networking hops. The accuracy of timestamping is one of the key building blocks for Ethernet switch devices intended for time-sensitive networks.

Figure 45: Timing and Synchronization



The IEEE 1588 Profile, IEEE 802.1AS, follows the same fundamental synchronization model as PTP where the transport technology and encapsulation is confined to Ethernet transport. IEEE 802.1AS profile has been expanded by the addition of parameters from IEEE 1588 that are required for use in automation networks.

For example, IEEE 1588 offers support for multiple time domains that can be synchronized in parallel. Accordingly, with IEEE 1588, network participants can be synchronized with a global time reference (like some atomic clock), as well as a second network time reference. This offers the option to use the global synchronization for unambiguous event logging, while the network-wide synchronized clock can be used for the Time-Aware Scheduler, since in this case, synchronization according to global conventions (such as the leap second) is not required.

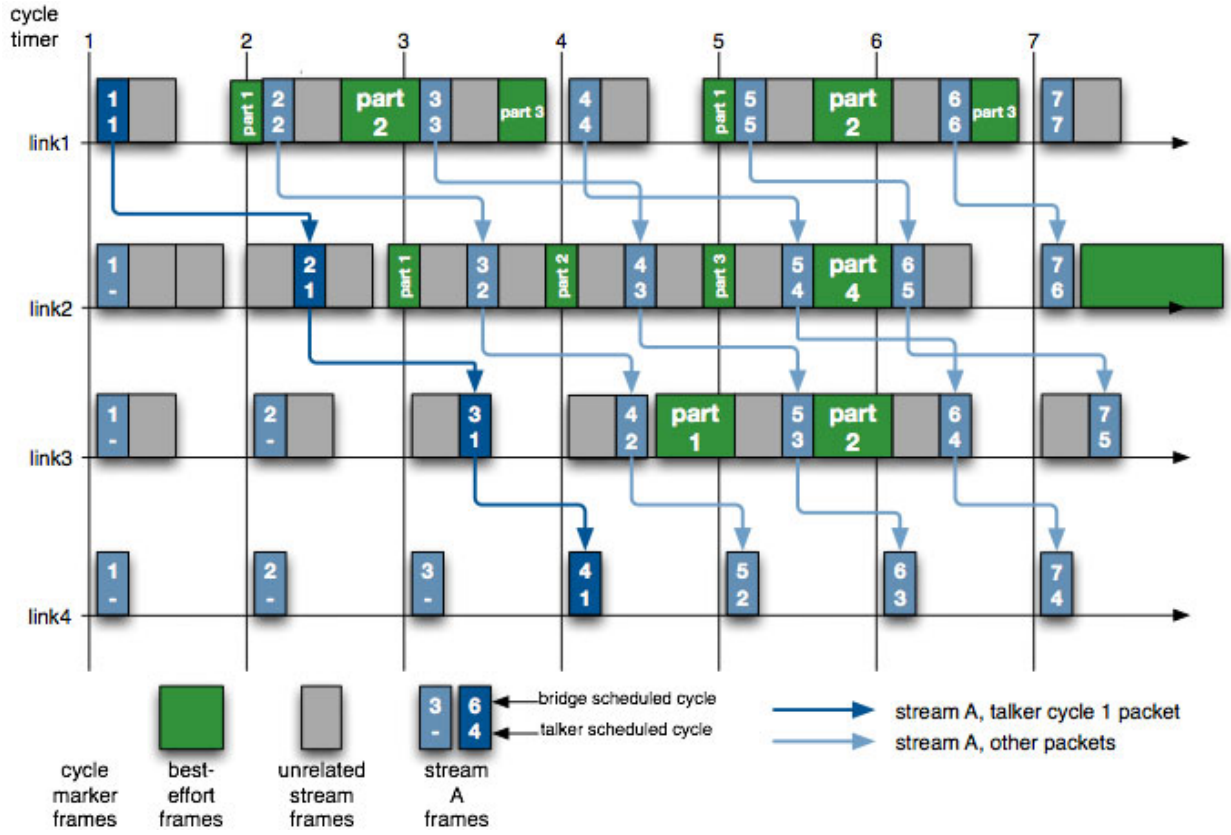
IEEE 802.1AS requires that all devices participate in BMCA, which is almost identical to default IEEE 1588-2008. The intermediate systems (bridges) has same performance as transparent clock for Ethernet. IEEE 802.1AS requires that all participating devices in the network are capable of supporting IEEE 802.1AS otherwise non-participating devices will block the time sync path from master slave. This is a single domain with Ethernet being the L2 layer, but support limited to a single port. With the enhanced revision (IEEE 802.1ASrev) it allows for all Ethernet (including link aggregation). This also supports multiple time domains. The one step transparent clock is a per hop behavior and hence both one-step and two-step transparent clocks can be present in the network. As a peer to peer Transparent Clock (TC), it has the ability to auto-generate a pDelay Response based on a pDelay Request. The correction field is updated to reflect the resident time.

6.1.4.2 Cyclic Queuing and Forwarding

The Cyclic Queuing and Forwarding method (defined in IEEE 802.1Qch) makes use of the mechanisms of the Time-Aware Shaper. However, when compared to the Time-Aware Shaper, this Traffic Shaper has significantly reduced requirements concerning the time precision of the transmission. As shown in Figure 41, the basic concept of the Cyclic Queuing and Forwarding method is to collect the data frames with reserved bandwidth received within a cycle and transmit them with higher priority at the start of the next cycle. Thus, the maximum end-to-end latency can be determined precisely through the number of hops on the transmission path and the configured cycle time.

This deterministic latency behavior will also scale with varying port speeds through intermediate node in the path. With these characteristics, Cyclic Queuing and Forwarding is well-suited to the sporadic data transmission of process. This functionality is realized by using a combination of Time Aware Policing (IEEE 802.Qci) and Time Aware Shaping (IEEE 802.1Qbv) which are both supported in this device. Both use the knowledge of the synchronized network time to police and schedule traffic through the switch. The policing avoids the queues being filled up with late or early arriving packets (outside the expected window), and scheduling ensures the accepted packets are forwarded at the start of every cycle. Additionally, the policing mechanics also allow for changing the assigned Queue at alternate intervals, which allows for the two queues to be setup and scheduled for transmitting at twice the cycle time. The cycle times can be determined based on the slowest switch latency for hop (port to port), across all the hops in a network. Configuring the time aware policer and scheduler for these cycles times will provide a very deterministic latency through the network.

Figure 46: Cycling Queuing and Forwarding



6.1.4.3 Time-Based Ingress Policing

IEEE 802.1Qci and IEEE 802.1Qch both extend the data structures developed for IEEE 802.1Qbv. IEEE 802.1Qci could be used to block excess traffic at the ingress side of a switch (for example, some node is malfunctioning) and preventing the offending traffic from entering to the rest of the network. The mechanism as defined in IEEE 802.1Qci allows discarding real-time data streams that use more than their reserved bandwidth, thus allowing the policing of streams. However, similar policing can also be achieved using conventional existing policing tools without requiring the rest of the TSN toolbox. The goal is mainly to protect for QoS and redundancy features against software bugs on endpoints or hostile devices, and also intermediate switches or bridges.

IEEE 802.1Qci relies on IEEE 802.1CB stream identification for its per-stream filtering and policing. The identified streams can be mapped to the available gates in the device, which open and close independently based on the calendar schedule. There are additional controls available at the gate, namely to limit the total numbers of bytes that are allowed through the gate during an open period. This is a per gate limit which is applied to across all the streams that are mapped to the gate. There is also a provision to switch queue assignment at the gate depending on the open or close state of the gate.

6.1.4.4 Time-Aware Scheduled Traffic

IEEE P802.1Qbv TAS adds a time-aware gate control list, which allows controlling the output gate open and close states associated with each queue according to a specific repeating schedule. Based on the schedule programmed in the gate control list, frames are selected for transmission. TAS allows building complex time-based schedules relative to a known timescale. As a result, TAS requires a fully synchronized and managed network to work. Each node in the network has to adhere to the network wide calculated schedules, which are known to be a real management burden for any larger and non-trivial topology. While TAS provides a deterministic time slot for a selected traffic (that is, a sender knows when it can transmit, and a receiver knows when to expect a certain type of traffic to arrive), a single interfering frame from another queue that is using the same egress port can break the schedule. Once the transmission of a frame has started, other traffic has to wait until the transmission has completed. This can cause the scheduled traffic to miss the schedule and add unwanted FDV. There are ways to mitigate the issue. For example, the schedules could contain a guard band period, which has to be the length of the largest possible interfering frame. No frame is scheduled during the guard band period, which ensures the egress port is always available for the scheduled traffic when its time slot starts. The obvious price to pay for the guard bands is the waste of bandwidth. The schedules would now contain periods when no traffic can be sent.

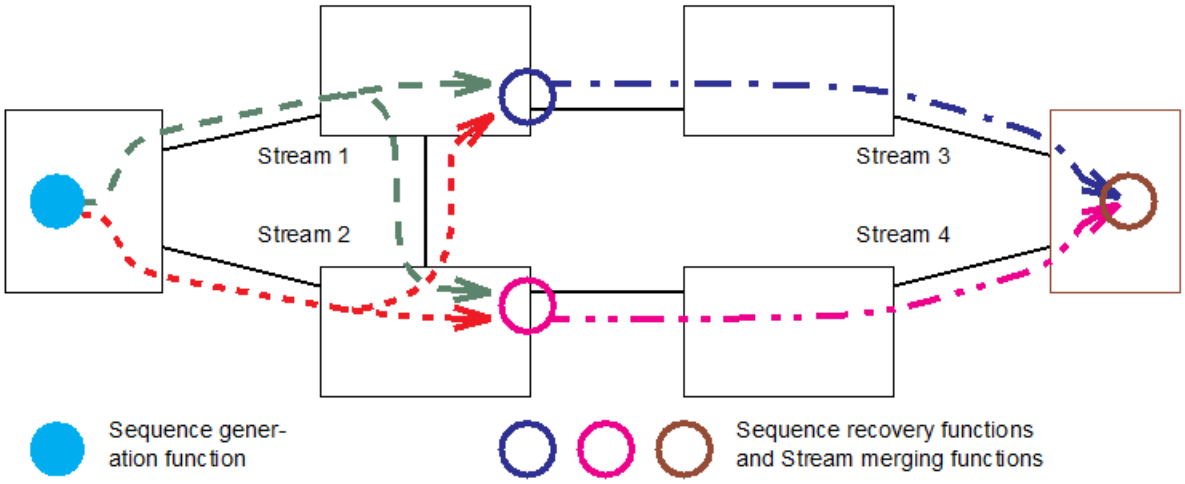
This device allows for time aware scheduling to function as an independent feature and can be effective in a time sensitive network with larger tolerance band (where the jitter due to interfering traffic is acceptable). For networks with tighter tolerance limits this feature can be combined with frame preemption.

6.1.4.5 Seamless Redundancy

In order to achieve seamless redundancy with IEEE 802.1CB, the Ethernet frames that need to be transmitted are replicated at the beginning of a redundant transmission path and subsequently forwarded through the network via multiple paths. Usually, the replication occurs either directly on the sending device or, if the end device does not support redundant network connections, such as the one illustrated in Figure 47, at the first network device on the transmission path. When the data arrives at the destination, the first redundant data packet is forwarded in the direction of the application layer. Packet duplicates received after the first packet are recognized via a redundancy field in the Ethernet header and discarded. Thus it is ensured that the redundant data transmission with IEEE 802.1CB is transparent for higher layers in the network stack and do not need to be taken into account.

In comparison to HSR (IEC 62439-3), PRP (IEC 62439-3), the redundancy mechanisms developed in the context of the IEEE 802.1CB offer the advantage that they can be used in any topology. Thus, IEEE 802.1CB is not limited to the otherwise absolutely required ring topology or topologies with completely independent networks. Additionally, IEEE 802.1CB is not restricted to exactly two redundant paths. To reduce the probability of packet loss, it is also possible with IEEE 802.1CB to utilize numerous redundant transmission paths with latency guarantees that are required by the application. Another important data point regarding IEEE 802.1CB is that its stream identification function is reused by other TSN standards. For example, an implementation of IEEE 802.1Qci relies on IEEE 802.1CB stream identification for its per-stream filtering and policing. The device supports hardware learning of streams for HSR and PRP.

Figure 47: Stream Duplication and Merging for Seamless Redundancy



6.1.5 TSN Applications to Industrial Ethernet

6.1.5.1 Overview of TSN Features

6.1.5.1.1 Time Synchronization

Time synchronization deals with frequency and phase synchronization of a global clock across all network elements in a time-sensitive network. A network element, which could be an Ethernet endpoint controller, bridge, switch, or router, is able to generate, propagate, recover or simply consume clock cycles depending on its role or position in a time sync network with precision and reliability. SyncE (ITU-T) and IEEE 1588 or IEEE 802.1AS define standards for transporting time (frequency or phase) reliably over Ethernet.

There are few key parameters which define quality of time synchronization:

- Precision of time synchronization – Applies to generation, recovery, propagation and consumption
- Ability to switch over to a new grand master clock in a short time (order of 100 ms)
- Allow for multiple grand masters at the same time – Provides redundancy and enables multiple time sync networks within physical network.

6.1.5.1.2 Cut-Through

Cut-through switching has been defined and supported in different network architectures to achieve low latencies through the switch and is a key feature of TSN where a high priority time sensitive packet is able to cut through the switch from ingress port to the egress port without being fully stored before forwarding.

6.1.5.1.3 Time-Aware Scheduler or Shaper (TAS)

A switch that supports a time-aware scheduler is able to schedule transmit of packets across the queues of a given egress port during a specified time window using a scheduler which runs on a global synchronized clock which spans across the time-sensitive network. Such a time aware scheduler enables transmission of time-sensitive packets across the network with minimal latency and jitter with the aid of the timing windows but requires that all the schedulers of network elements be configured and managed centrally. TAS is being defined by IEEE standard IEEE 802.1Qbv and is analogous to the AVB credit based shaper defined by IEEE 802.1Qav standard but instead of using credits, the shaper uses time gating to shape or multiplex traffic across the queues of a given port. Requirements for configuration and management of time-aware scheduler across multiple network elements are being defined as part of IEEE 802.1Qcc.

6.1.5.1.4 Seamless Redundancy

Seamless redundancy deals with setting up and managing redundant paths within the time-sensitive network in order to provide high availability. This is a critical feature in TSN for reliable distribution of time as well as forwarding of high priority time-sensitive traffic through the network.

From the standards perspective, IEEE 802.1Qca deals with explicit path control, bandwidth and stream reservation, redundancy for data flows and distribution of control parameters for time synchronization and scheduling. IEEE 802.1CB deals with the standards associated with replication of frames and elimination of redundant frames for bridges and end points. IEEE 802.1CB is an alternative to existing protocols, Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR).

6.1.5.1.5 Time-Based Ingress Policing

Time-based ingress policing is an important feature in a TSN to ensure frames belonging to a stream are queued in a network element in a specified time window. This feature along with preemption, cut-through forwarding, and time-aware scheduler help achieve time-deterministic network.

IEEE 802.1Qci deals with frame counting, filtering, and policing and service class selection for a frame of a specific data stream to support time-based ingress policing.

6.1.5.1.6 Centralized Network Configuration

Centralized network configuration is a key feature of TSN which deals with configuration of network elements (bridges, switches, and routers) in TSN to support various features such as:

- Cut-through (enable or disable)
- Time-aware scheduler
- Seamless redundancy
- Time-based Ingress policing

A centralized network configurator as part of TSN remotely manages network elements supporting these features.

Requirements and parameters of a centralized network configuration entity are covered by IEEE 802.1Qcc.

6.1.5.2 TSN Requirement in Industrial Applications

Industrial Ethernet uses standard Ethernet media to support existing open or proprietary protocols such as Ethernet or IP, and replace widely used fieldbus technology. Some of the advantages for deploying Ethernet based networks are as follows:

- Increased speeds and distance
- Ability to use standard access points, routers, switches, cables, and optical fiber
- Ability to have more than two nodes on a link
- Better interoperability
- Elimination of separate commissioning interface for data traffic
- IT integration into devices

While Ethernet has been adopted for time-noncritical industrial applications for some time, there were issues with using the technology for time-critical applications due to inherent reliability concerns and non-deterministic latencies through the network. This has changed over the past few years with the enhancements in Ethernet Standards, for example, IEEE1588 and AVB or TSN, which enabled further penetration of Ethernet technology in almost every application of industrial automation.

Industrial automation environments consists of distributed control systems containing a large number of interconnected devices that exchange data through communication networks. The system implements closed control loops between the controller and the devices such as actuators and sensors. Table 6 captures various service types associated with the traffic types in Industrial Ethernet applications.

Table 6: Traffic Classification in Industrial Ethernet

Service Types	Traffic Types	Status
Best Effort	Configuration	Preemptable Traffic
	Diagnostic	
	WEb Services	
	Events	
Real-time (AVS Stream) Control Data Traffic	Real-time Diagnostic	Express Traffic
	Automation	
	Motion Control	

6.1.5.2.1 TAS in Industrial Automation Applications

Using Time Aware Scheduling (TAS) at each node of the network, including devices and controllers, can exchange information in specified timing windows across the network. The objective of TAS is to keep the window as narrow as possible so there is more time available for processing.

The control loop or control cycle is divided into two parts:

- Communication cycle
  - Time during which controller and the devices exchange data
  - Requires deterministic packet scheduling
- Compute cycle
  - Time during which controller processes the data
  - No time sensitive data is exchanged and is ideal for best effort traffic

In a given Industrial Automation environment, there can be multiple such control loops which are running concurrently.

The following table captures typical device and network level requirements for supporting Industrial automation applications. The requirements of a device such as an Ethernet switch can vary depending on the number of such devices in the network and where it is deployed. It is more important to understand the requirement for a given application, estimate number of hops in a given network, and provide budget for each node in the network.

Example:

A device which is used at multiple hops inside a network needs to have latency of no more than 1.5 μs, while a device such as BCM56070 can exceed that latency since there can be only one such device in the network.

The requirements also change and become more aggressive with time, as applications require more efficiency and link utilization from infrastructure deployed.

6.2 Time Aware Scheduling

TAS is supported only for industrial part numbers.

6.2.1 Time-Sensitive Traffic

Conventional Ethernet networks do not guarantee a timely end-to-end delivery of frames due to contention at different nodes in the network. When a network is used for a combination of traffic profiles, including time-sensitive (real-time) traffic as well as non-time sensitive Ethernet data traffic, it is required that the time-sensitive traffic goes through the network with minimum and deterministic delay while data traffic uses the remaining bandwidth maximizing the link utilization.

When both time-sensitive traffic as well as Ethernet data traffic flows through the network, the time-sensitive traffic will be delayed by two categories of traffic flows:

- Ethernet data traffic flows which create contention at the Egress port where time-sensitive (express) flows are delayed while a large Ethernet data frame is being transmitted
- Other time-sensitive traffic flows from different combinations of sources and destinations

Either of these cases of contention can cause significant delay as well as unpredictable jitter which make the network unsuitable for time-sensitive applications. The delay to the time-sensitive frame is dependent on port speed and the size of the Ethernet data frame. Ethernet data frames can vary in size and cause variable delays.

In case when there is contention of two time-sensitive traffic flows at an Egress port, both flows will be subjected to random delays even if the frames are smaller.

Through Time Aware Scheduling (TAS), protected time windows are created for time-sensitive traffic leaving unprotected windows for Ethernet data traffic. In industrial applications, it is possible for multiple control loops to be supported through a single port in which case multiple protected windows need to be created in order to avoid traffic belonging to each control loop contending for transmission at a given port.

At every egress port, whether it is in a switch or an endpoint controller, the traffic is shaped or scheduled per queue such that transmission occurs only during specific time window designated by central network configuration entity. With time-aware traffic going through dedicated high priority queues during dedicated time windows for transmission and best effort Ethernet data traffic going through low priority queues, contention can be completely avoided making delivery of time-sensitive frames fully deterministic.

6.2.2 TAS Overview

TAS functionality is distributed primarily in two functional blocks:

- TAS state machines
- Scheduler

6.2.2.1 TAS State Machines

This block includes the following features:

- Implements standards-defined state machines and configuration data bases with some differences.
- Receives PTP time from IP block and adjusts it locally to establish a common reference across the system which is used for all configuration related operations.
- Implements all state machines in time sync clock (TS_clk) domain and transfers OperGateStates and holdRequest command through an asynchronous CDC FIFO into core clock (core_clk) domain for consumption of scheduler and other logic.
- Implements administrative and operational databases using two Ping-Pong buffer structures along with OperDataBasePointer, which points to operational database while other database serves as the AdminDataBase. When the administrative database has to be copied to the operational database, the function is achieved by simply switching the pointers instead of copying the databases.
- Supports two modes of operations:
  - PTP_mode – Indicating that the PTP time is synchronized across network and all configuration state machines need to work as specified in the TAS standards specification.
  - Non-PTP_mode – Indicating that the PTP time is not synchronized across the network and that TAS state machines operate in proprietary way.
- Supports generating holdRequest as specified by operation configured in OperControlList table.
- Supports only specifying AdminBaseTime configuration in the future to simplify the implementation of state machines. API layer has to take care of configuration associated with AdminBaseTime in the past.
- Detects and reports exceptions associated with configuration of administrative base times to software and moves the state machines into error states, stopping gate operations as specified in the administrative or operation databases.

6.2.2.2 Scheduler

The BCM56070 Scheduler supports the following features:

- TAS features are overlaid on legacy features supported by MMU.
- The scheduler supports gate control operation as per TAS requirement.
- The credit-based shaper is enhanced to support TAS requirement where credit refresh happens only when gate is in open state.
- Express traffic is enqueued to high priority queues which inherently means queues carrying express traffic will have higher priority over queues carrying preemptable traffic if gates are open for these queues.
- A CoS queue participates in scheduler arbitration only if the gate of that queue is in open state.

6.2.2.2.1 Special Encoding of TimeInterval

BCM56070 implements two mechanisms to start a new cycle based on the value of TimeInterval field of last entry of GCLT:

- TimeInterval = 0x000 (for cases where PTP is not active) – In this case, LESM state machine restarts from the first table entry
- TimeInterval = 0xFFF (for cases where PTP is active) – In this case, LESM state machine waits at the end of cycle until CycleStart operation is initiated by Cycle Timer State Machine.

NOTE: This replaces that need for END_OF_CYCLE state specified in standards defined LESM.

6.2.2.2.2 Sizing of Gate Control List Table (GCLT)

The TAS scheduler at a given port has to be able to support multiple control loops being executed at network level. Each loop can use a different path in the network, but a given port in the device will have to support these control loops in a time-division multiplex (TDM) fashion within an operational cycle. In addition, each control process may have different sampling period.

The number of entries required for each operation may also go up if phold_req has to be set or released before or after gate states are changed.

In the BCM56070 implementation, GCLT (OperConfigTable) is used to support multiple control loop operations through a given port.

6.2.2.2.3 Jitter Due to Async FIFO and Port Level TDM

TAS state machines that operate with reference to the global clock are implemented in the TS_clk domain. The scheduler arbitration and entire buffer management function are implemented in the core clock domains. The transfer of gate states and phold_req signals from the TS_clk to core_clk domains happen through an 8-deep asynchronous FIFO. The asynchronous FIFO introduces some delays (or lag) due to the synchronization requirements.The asynchronous FIFO also introduces jitter. The lag and jitter depend on the TS_clk cycle time and core_clk cycle time and need to quantified and budgeted into over-device-level jitter specification.

In addition, the BCM56070 implements an egress port scheduler that multiplexes data from each port in a TDM fashion. This means the transmission of data from a given queue of a given port will start only after the port is selected by the egress scheduler. This process adds additional jitter with reference to the gate open or close operations that originate in the TS_clk domain.

6.2.2.2.4 Minimum TimeInterval Requirement

TimeInterval in a given entry of GCLT table can be specified in the order of nanoseconds. Given the port level scheduler works in a TDM fashion, a given port may get picked at least once in TDM calendar whose length can range from 200 ns to 1 μs, depending on core_clk, number of ports, and port speed. With these parameters and constraints, a meaningful TimeInterval setting needs to be calculated per port and can be in the order of microseconds.

6.2.2.2.5 Gate States Selection

There are three sources of gate states which drive the gate of each queue, allowing it to participate in scheduler arbitration. In addition, to support flush operation in TAS based scheduler, the GateState needs to be overridden to all gate open states when the flush operation is in progress in order to enable flush operation.

6.2.2.3 Credit-Based Shaper

Shaper requirements for TAS are overlaid on the legacy AVB shaper implementation. The credit-based shaper block within the MMU has the following features:

- Supports AVB shapers

  In token bucket shaper, when a queue becomes empty, tokens are reset to zero
- Additional mode for TAS

  Option to freeze a shaper tokens when the gate is closed:
  - When a queue's gate is closed and no packet is scheduled from queue, no more tokens are added to the queue's bucket
  - This option controls burstiness of unscheduled queue to make sure they do not burst when their gate is opened

A minimum shaper is not recommended when TAS mode is enabled

The BCM56070 implements per queue Shaper_Configuration_Mode to support both legacy AVB and TAS specific shaper modes.

6.3 Seamless Redundancy

Seamless redundancy is supported only on the industrial part number.

6.3.1 Overview

Seamless Redundancy (SR) is a function that ensures almost zero packet loss in presence of single failure in the network such as node failure, link failure, link error, and congestion loss.

The SR function operates by duplicating each packet on two disjointed paths toward the destination. The duplicate packets will have the same sequence number encoded in a SR-Tag. At the destination, the first packet from the duplicate that is received in fault-free state is accepted, and the second packet from the duplicate is dropped. This ensures that even when one packet is received, the destination has not lost any packets.

SR is implemented per the following three different standards:

- High-availability Seamless Redundancy (HSR)
- Parallel Redundancy Protocol (PRP)
- IEEE 802.1CB Standard for Seamless Redundancy

The functionality of these three standards is almost the same, but they differ on the format, encoding, and placement of the SR-Tag in the packet. For example, the SR-Tag is placed right after the Ethernet header in cases of HSR and IEEE 802.1CB standards, while it is placed before the Ethernet FCS in PRP standard.

The granularity of SR flow is different in these three standards. HSR and PRP have low granularity, where an SR flow is only based on MAC-SA, while IEEE 802.1CB has high granularity and its SR flow is based on L2, L3, or L4 octuplets.

In addition, PRP is mostly defined as P2P technology between two PRP nodes, while HSR and IEEE 802.1CB are defined mostly for Ring topology. However, HSR and IEEE 802.1CB can have a mesh, star, or shared-bus topology.

SR in HSR, PRP, and IEEE 802.1CB are only defined for L2 switching and are not defined for L3 routing.

NOTE: The source and destination may be on different SR networks with different SR technologies.

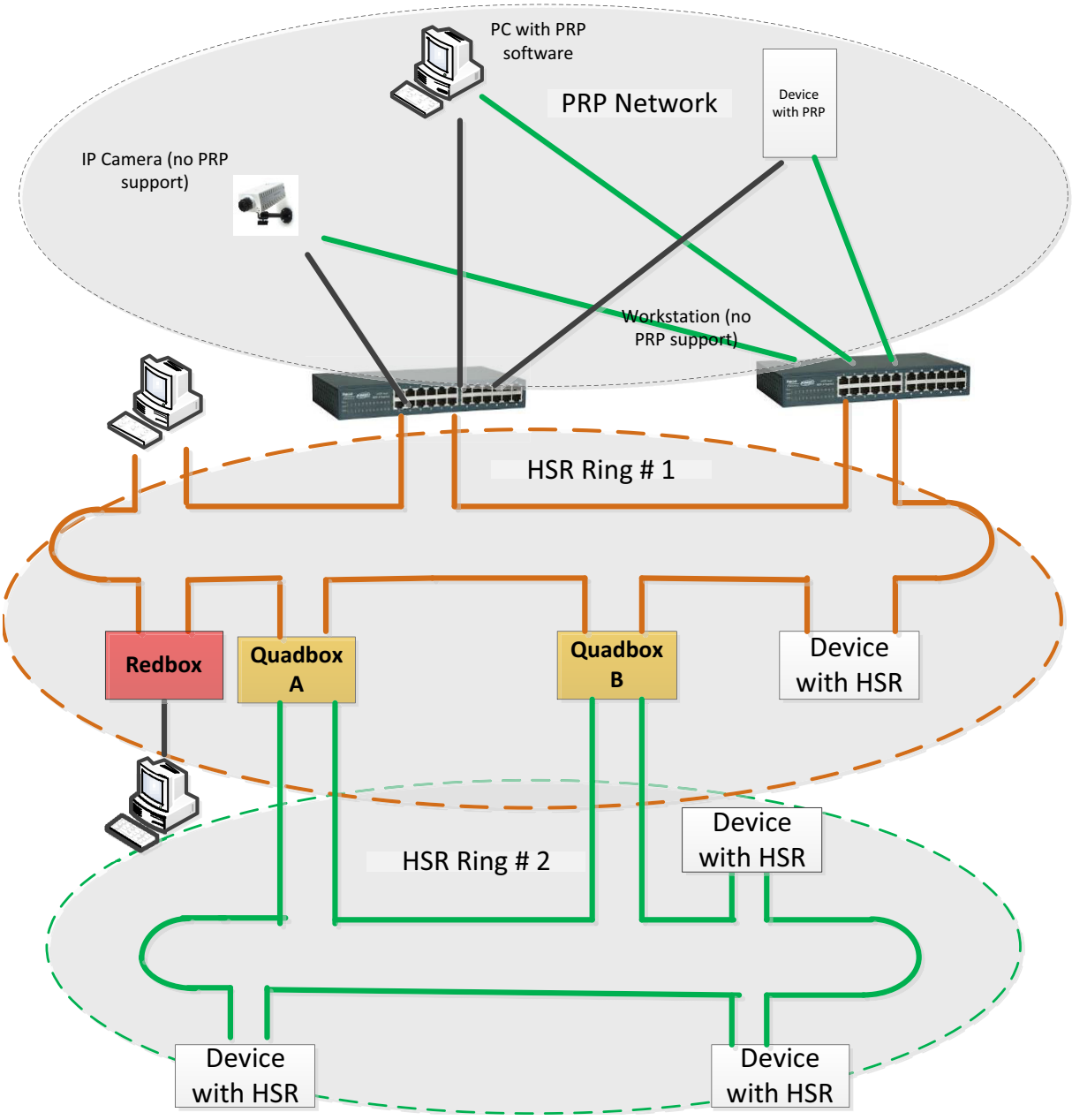
An example is shown in Figure 48. In this figure there is one PRP network and two HSR ring networks. The PRP network is connected to HSR Ring 1 by two interworking switches. The packets transiting between HSR and PRP maintain their sequence number, and just the SR-Tag is translated between the two networks. Based on the standards, up to seven PRP networks can be attached to a single HSR Ring.

Figure 48 shows interworking between HSR Ring 1 and HSR Ring 2 through two Quad-boxes, A and B. The SR-Tag of packets transiting between the HSR Rings is unchanged.

Figure 48 also shows a concept called *Red-box*. Red-box is a node that behaves as proxy to a terminal or device and creates and terminates SR duplicate packets on behalf of the device or terminal. For the most part, the BCM56070 functions either as a Red-box, Quad-box, or Interworking switch in the SR networks.

IEEE 802.1CB is similar to HSR, with the main difference being that the IEEE 802.1CB flows are much more granular than HSR, and are based on L2, L3, or L4 octuplets (DA, VLAN, PRI, SIP, DIP, Protocol, S-Port, and D-Port).

Figure 48: PRP and HSR Interworking



6.3.2 SAN, DAN, and LRE

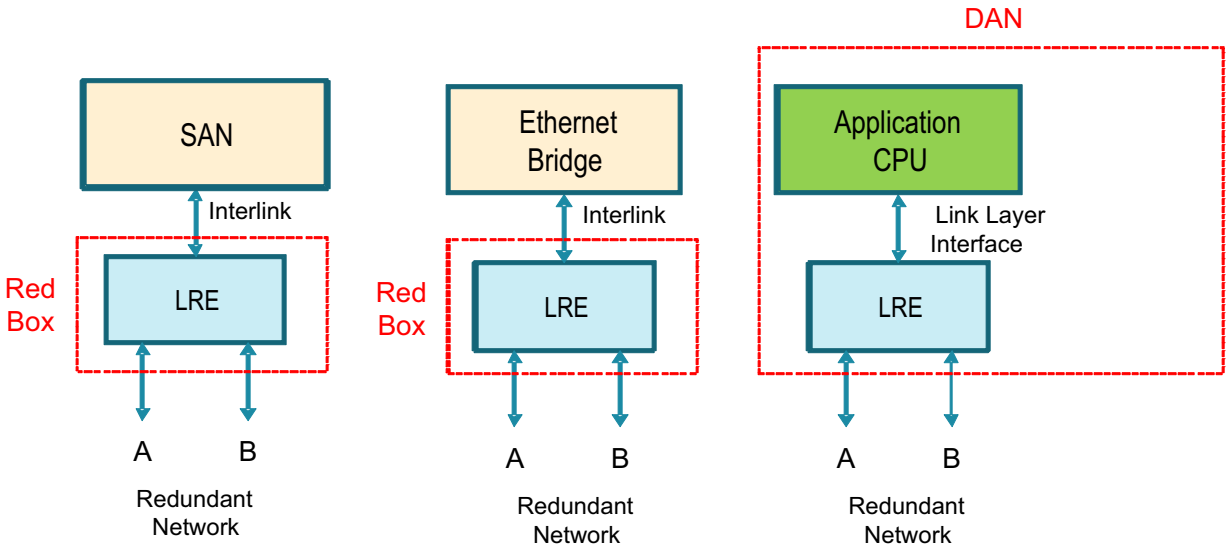
Link Redundancy Entity (LRE) a logical three-port entity that connects a non-Redundant Network to a Seamless Redundant Network. One port is connected to the non-Redundant network and two ports are connected to the redundant network.

A Terminating node (such as a computer or a sensor) that comprises of LRE is called a Dual-Attached Node (DAN), since it connects to the network using two ports. The application interface to the LRE is through the Link Layer Interface, which can be virtual or physical (such as PCIe).

A Terminating node that does not comprise of LRE is called a Single Attached Node (SAN). A SAN has a single port. A SAN can be connected to a redundant network through a Red-box. The following figure shows two Red-boxes and a DAN. The Red-box uses Interlink to connect to either a switch or a SAN, and its job is to connect a non-redundant entity to a redundant network.

NOTE: Red-boxes also include the Application Layer connected to the LRE through the Link Layer Interface.

Figure 49: Link Redundancy Entity



The port that is connected to non-Redundant network can be a physical port, a logical port, or internal port inside a switch. When the non-Redundant port is connected to an IEEE 802.1 switch, it is called an *Interlink*. When the non-Redundant port is connected to a higher-level protocol that is processed by a CPU, it is called a *Link Layer Interface*.

The main function of LRE is to duplicate packets received from a non-Redundant port and send duplicate packets to the two ports connected to the redundant network. In the reverse direction, the LRE receives duplicate packets from the Redundant network, forwards only one copy to the non-Redundant network, and discards the duplicate packet.

PRP, HSR, and IEEE 802.1CB support the LRE function.

6.3.2.1 Red-Box

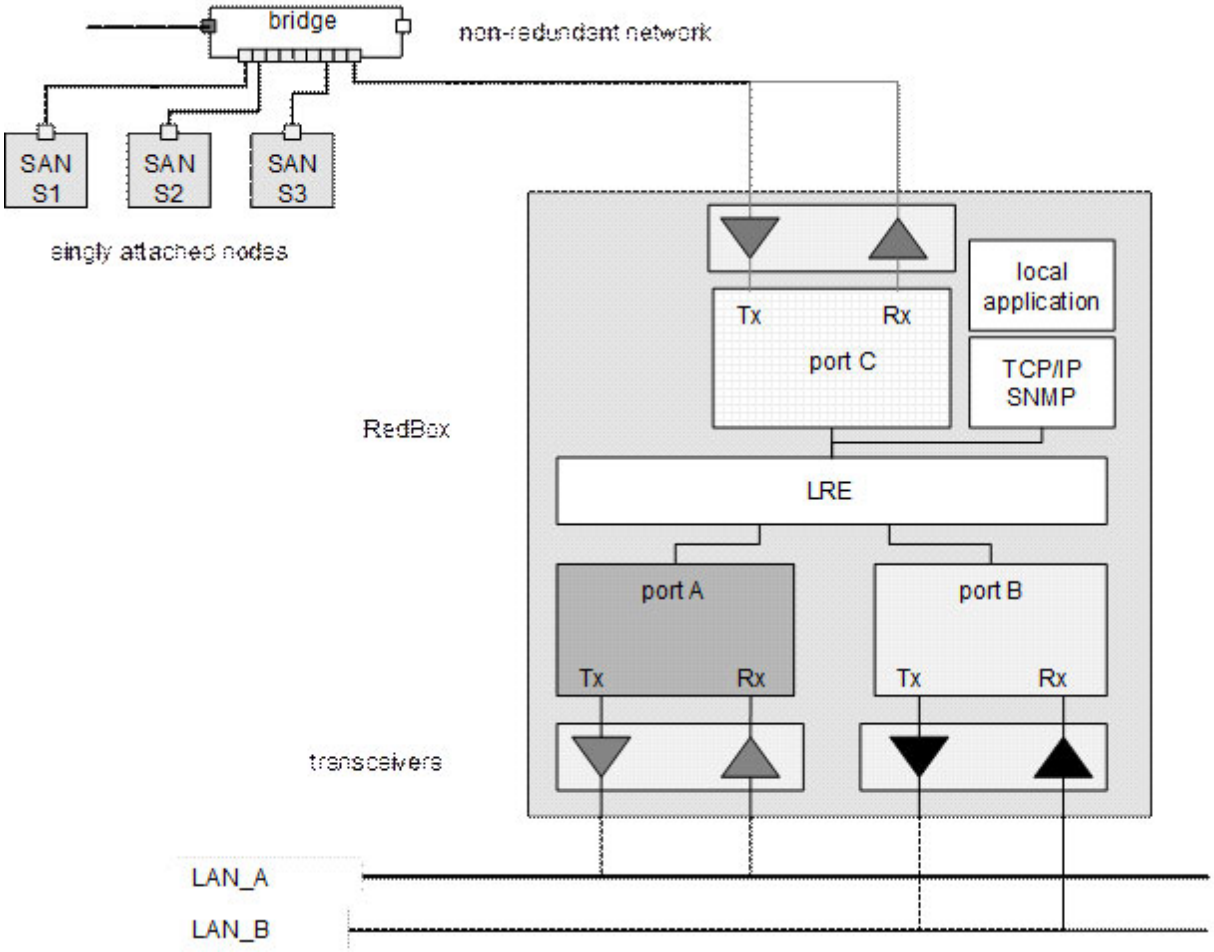
Red-boxes connect Single Attached Nodes (SAN), such as sensors or IEEE 802.1 bridges that do not have SR capability, to a Seamless Redundant Network. The Red-box mimics the SANs connected behind it (called VDAN or virtual DANs) and multicasts supervision frames on their behalf. The Red-box is itself a DANP and has its own IP address for management purposes, but it may also perform application functions.

6.3.2.1.1 PRP Red Box

Figure 50 shows the model of a PRP Red Box with three interfaces:

- Port A
- Port B
- Port C (Interlink)

Figure 50: PRP Red Box



The Interlink is used to connect the Red-box to a switch or a SAN. A local Application Layer may also exist in the PRP Red-box. Such an Application Layer may be connected to the Red-box through the same Interlink.

The flow of traffic in a PRP Red-box is only between PRP ports (A or B) and the Interlink. There is no traffic forwarding between PRP ports (between A and B).

6.3.2.1.2 HSR Red-Box

An HSR Red-box is similar to a PRP Red-box, with the exception that HSR Red-box allows traffic flow between HSR ports (A and B) in order to be able to support Ring topology. An HSR Red Box can also be connected in non-Ring topology, such as connecting to two LANs, similar to PRP. In this configuration, the traffic forwarding between the HSR ports can be disabled.

The HSR Red Box has a LRE that performs the following duties of the HSR protocol:

- Forwards the frames received from one HSR port to the other HSR port, unless the frame was sent already.
- Receives frames addressed to its own upper protocols.
- Prefixes the frames sent by its own upper layers with the corresponding HSR tag before sending two copies over its HSR ports.

An HSR Red-box can be operated in one of three modes:

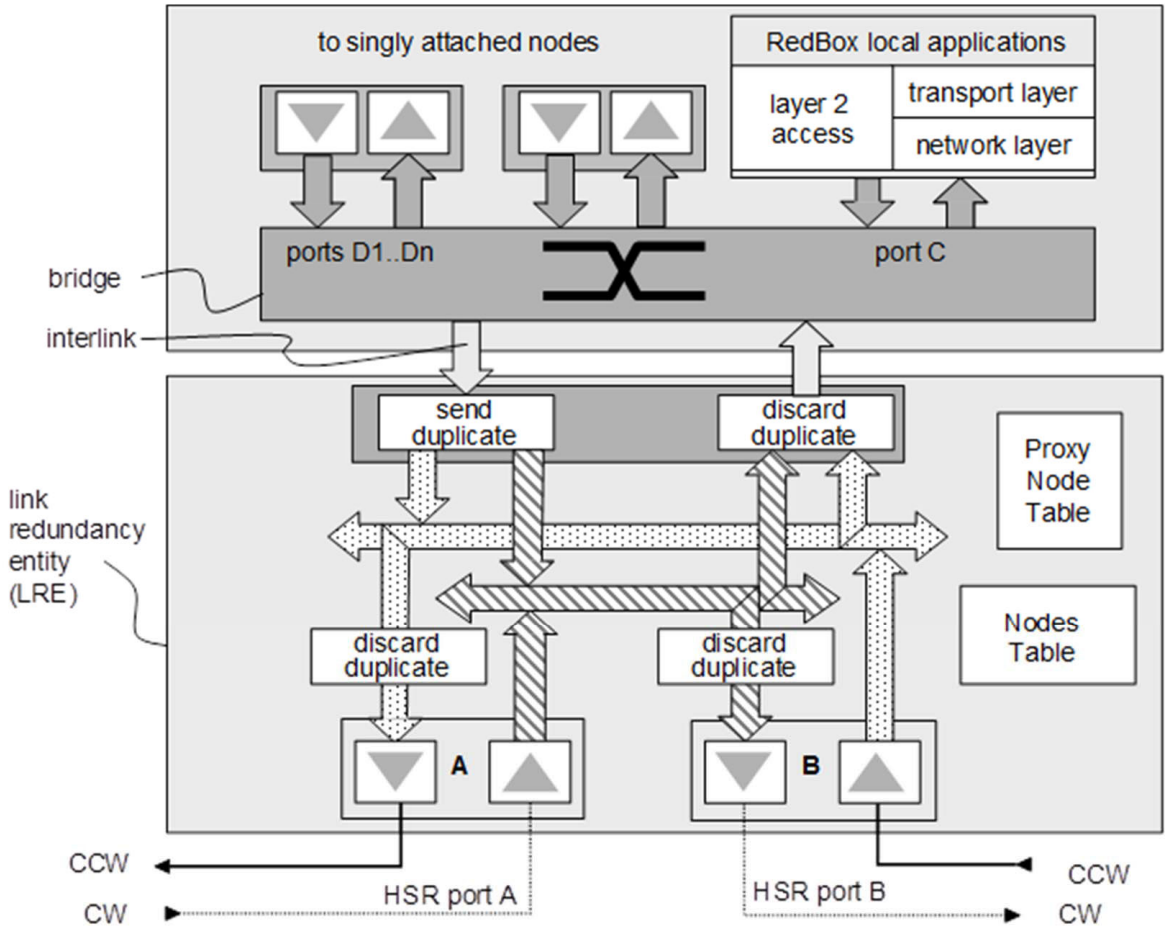
- HSR-SAN mode (Connects a SAN to HSR network).
- HSR-PRP mode (Connects a PRP network to HSR network).
- HSR-HSR mode (Quad-box, connects two HSR nodes or networks).

Depending on the mode of operation, the frame handling at the interlink interface of the Red-box differs. The HSR-HSR mode connect HSR rings to each other. It can also function as an HSR Relay in a mesh HSR network, where an HSR node is connected to three other HSR nodes and forwards the packet between those HSR nodes. Even in HSR-HSR mode, the HSR Red-box discards duplicate packets.

The switching logic in Figure 51 can be incorporated into the Red-box, so interlink becomes an internal connection.

A simple Red-box is present in every node because the LRE makes a transition to a single non-HSR host. Also, it is common to have more than one host in a node because a port for maintenance often exists.

Figure 51: HSR Red Box



6.3.2.2 IP and MAC Addresses

Both ports A and B of a Redundancy Node should be configured with the same MAC address. This address should be unique in the network. Also, MAC learning should consider the redundant port A and B as a single logical port.

Network supervision uses multicast packets. All nodes in the network are configured to operate with the same multicast address for network supervision.

The IP address or addresses of any node or bridge (LAN_A plus LAN_B) are unique within the whole network. A device may have several IP addresses.

A Dual Attached Node (DAN) has the same IP address or addresses when seen from either LAN_A or LAN_B. Bridges on LAN_A and LAN_B are considered as SANs and have different IP addresses for network management, even if they occupy the same location.

6.3.2.3 TSN and Seamless Redundancy

This section highlights the relationship between the TSN streams and Seamless Redundancy (SR) flows.

6.3.2.3.1 TSN Stream

A TSN stream is a stream that requires stringent QoS treatment in the network, and consequently in the BCM56070, such as preemptive queuing, cut-through switching, TAS.

A TSN stream is classified based on (DA, VLAN, and PRI). Any MAC-DA (unicast or multicast) can be used, and there is no reserved MAC-DA for TSN streams.

TSN streams require an ingress maximum transmission unit (MTU) check and ingress metering to ensure their stringent QoS.

6.3.3 Packet Formats

Figure 52 and Figure 53, respectively, illustrate PRP and HSR packet frame formats.

Figure 52: PRP Frame Format

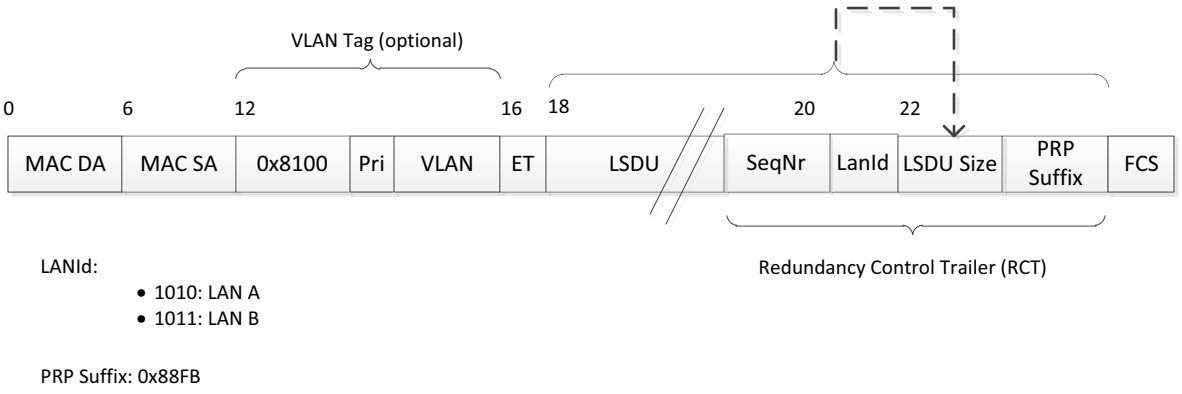
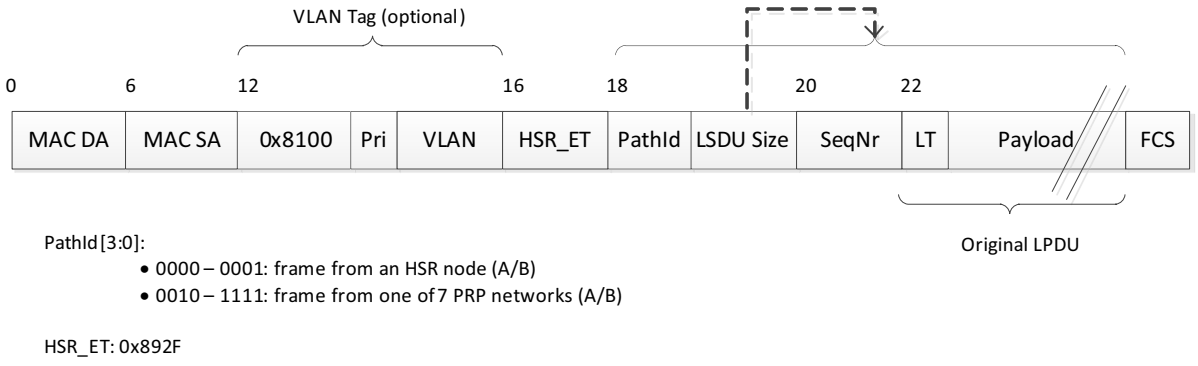


Figure 53: HSR Frame Format

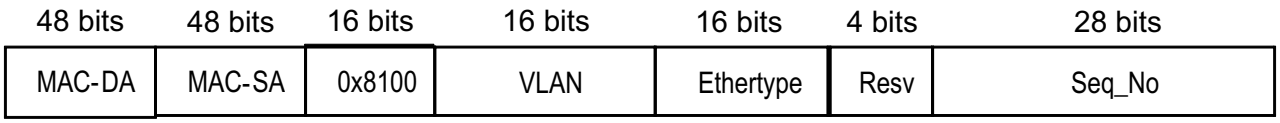


NOTE: In the HSR tag, the Path_ID is a 4-bit number that consists of a NET_ID (3 bits) and LAN_ID (1 bit).

- LAN_ID indicates to which of the two HSR ports the packet is sent.
- NET_ID indicates which network generated this packet. If the packet is generated by the HSR network, then NET_ID=0; otherwise, if the packet is generated by a PRP network and then enters the HSR ring, then NET_ID = 1–7 indicates which one of the seven PRP networks the packet was originated from.

The following figure shows the frame format currently defined in the IEEE 802.1CB draft standard. As the following figure shows, the IEEE 802.1CB tag is a 32-bit tag that contains only a sequence number. The NET_ID is based on the VLAN tag of 802.1CB packets and is not carried in the IEEE 802.1CB tag. Additionally, the unicast MAC-DA addresses can be from a specific multicast MAC address range defined by IEEE 802.1CB.

Figure 54: IEEE 802.1 CB Frame Format



6.3.3.1 PRP and HSR Supervision Frames

Each DAN multicasts a PRP or HSR supervision frame over both of its ports with the format specified in every LifeCheckInterval (default is a 2-second interval). It is assumed that the CPU (embedded or external) will generate and process PRP supervision frames. It is recommended to use IFP to filter supervision frames to the CPU.

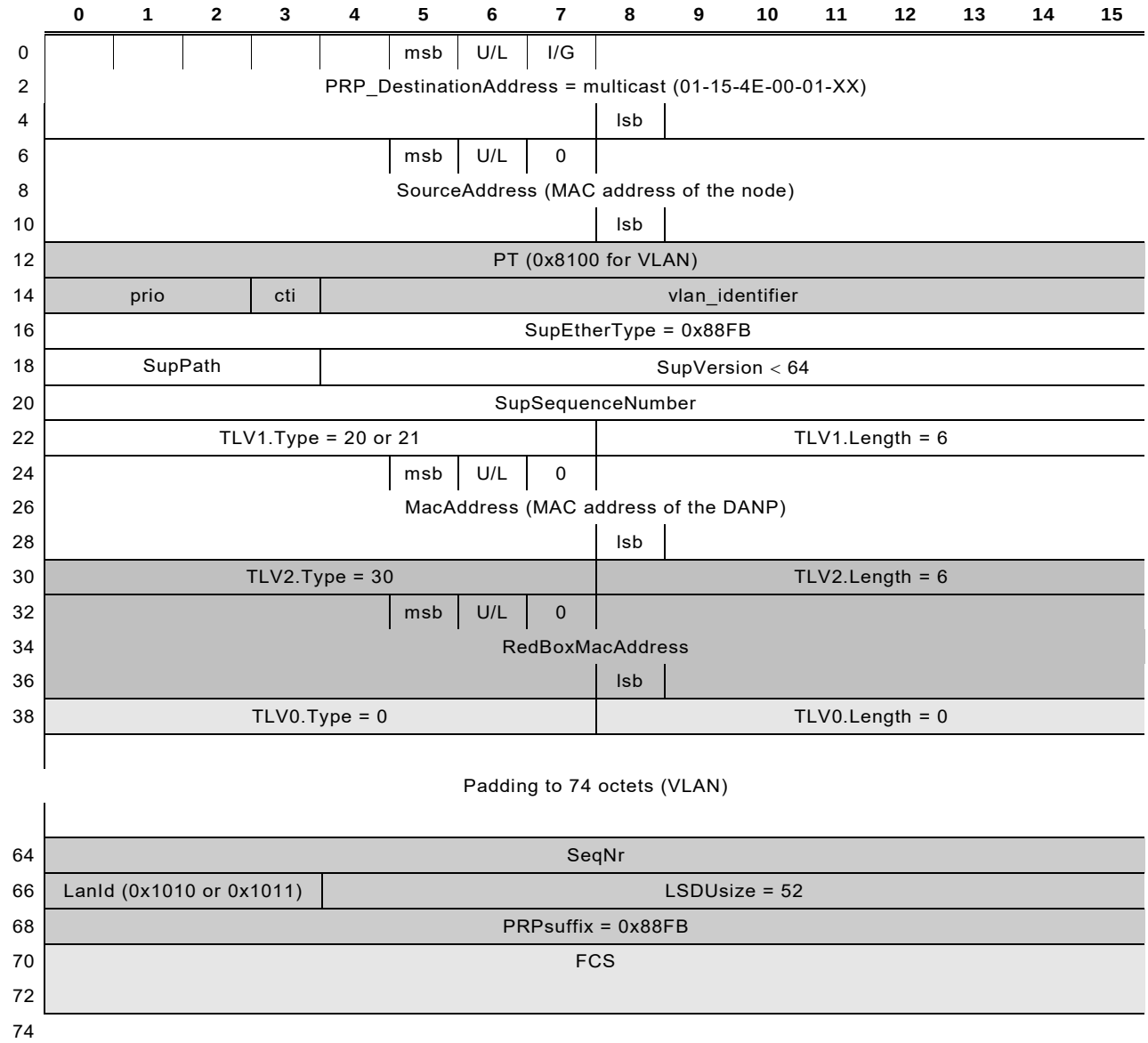
6.3.3.1.1 PRP Supervision Frame

The following figure shows a PRP supervision frame with a VLAN tag. As the figure shows, these frames are identified by their MAC address and EtherType:

- MAC = 01-15-4E-00-01-XX
- EtherType = 0x88FB

The PRP Supervision Frame may also be without a VLAN tag.

Figure 55: PRP Supervision Frame

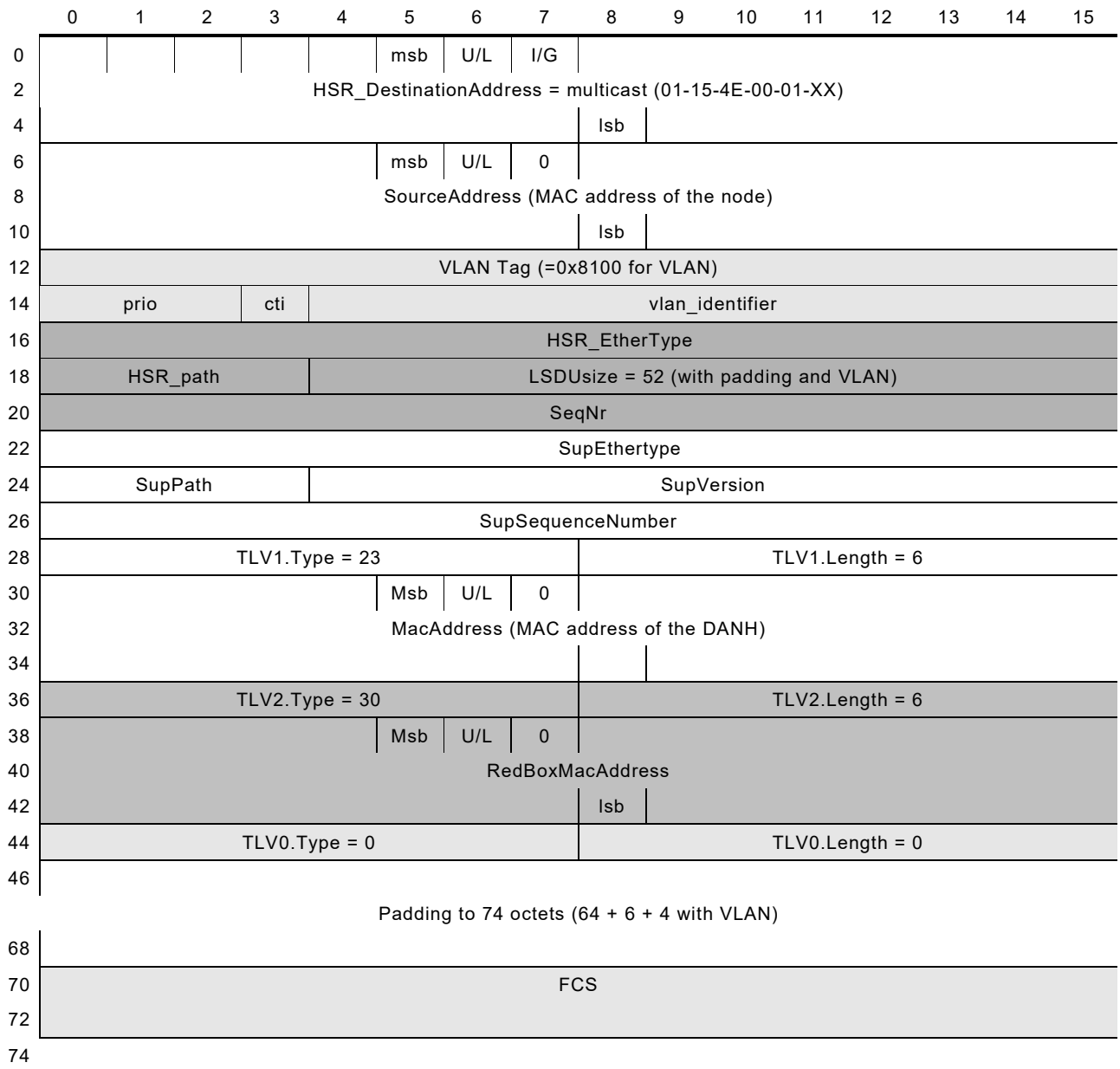


The following figure shows an HSR supervision frame with a VLAN tag. The HSR supervision frame may also be without a VLAN tag.

The HSR supervision frame is identified by the following:

- MAC = 01-15-4E-00-01-XX
- EtherType = 0x892F

Figure 56: HSR Supervision Frame



6.3.4 Seamless Redundancy Functions and Features

This section describes the SR functions and features.

6.3.4.1 HSR

HSR functions and features are as follows:

- Red-box
  - Mode H
  - Mode N
  - Mode T
  - Mode M
  - Mode U
- HSR-HSR Quad-box
- Relay box
- Flow ID
  - SA based (4K flows)

6.3.4.2 PRP

PRP functions and features are as follows:

- End-node
- Red-box
  - Duplicate Discard mode
  - Duplicate Accept mode
  - Transparent mode
- PRP-HSR Red-box
  - Support up to seven PRP domains
- Flow ID
  - SA based (4K flows)

6.3.4.3 IEEE 802.1CB

IEEE 802.1CB functions and features are as follows:

- Red-box
- Relay box
  - When all three ports of the node are 1CB
- Flow Identifier (see Table 7)
  - SA (4K flows)
  - SA, PRI (4K flows)
  - DA, VLAN, PRI (4K flows)
  - DA, VLAN, PRI, SIP, DIP, Protocol, S-Port, D-Port (1K flows, VFP-based)
- Common requirements (HSR, PRP, 1CB)
  - Fully configurable forwarding logic (to enable any forwarding behavior)
    - Configurable Ingress Logic (via IFP and configurable logic)
    - Configurable Egress Logic (via EFP and configurable logic)

- Sequence number
  - 16bit sequence number for PRP, HSR, 1CB
- Sliding window (configurable per-flow)
  - 128 bit (4096 flows)
  - 256 bit (2048 flows)
  - 512 bit (1024 flows)
  - 1024 bit (512 flows)
  - 2048 bit (256 flows)
  - 4096 bit (128 flows)
  - Optional: Automatic sliding window size adjustment (BRCM proprietary)
- Acceptance Window (configurable per-flow)
  - 2^n, where n = 5 to 16
- Sequence Window Aging
  - AGE_OUT_Time × 2^n, where n = 0 to 15
  - AGE_OUT_TIME (16 bits) in ms
- MAC Entry Aging
  - Configurable as per standard switch
- WRONG_LAN Aging
  - Same as Sequence Window Aging
- MTU size check
  - Per Port
  - Per TSN-Circuit
  - Per SR-flow
  - Per Egress queue
  - 32 MTU profiles. Each stream can select one profile
- Shared VLAN Lookup
  - Needed for 1CB (each path uses a different VLAN)
- Modified SrTCM metering for TSN streams
  - Separate architecture specification
- Management Information Base (MIB) counters per HSR, PRP, or 1CB standards
- Control or Management packet redirection to CPU
- Software MAC and flow learning
- Hardware and software Sequence Number window reset
- Cut-through forwarding for HSR and IEEE 802.1CB
- Queue Assignment
- Per-TSN-stream INT_PRI assignment
- Per-SR-flow INT_PRI assignment
- Support IEEE 1588v2 forwarding
  - Requires forwarding duplicate and non-duplicate PTP messages to CPU
  - Requires supporting 1-Step and 2-Step TC
- Ingress and egress mirroring
- Hardware MC learning for HSR or PRP

6.3.5 Network-Level Flow

This section describes the unicast and multicast network-level flow in HSR, PRP, and 802.1CB networks.

Table 7: Flows Supported in HSR, PRP and IEEE 802.1CB

SR Type	Flow Definition	Number of Flow[a]
PRP	MAC-SA	4096
HSR	MAC-SA	4096
IEEE 802.1CB	MAC-SA	4096
IEEE 802.1CB	{MAC-DA, VLAN, Pri}	4096
IEEE 802.1CB	{MAC-DA, VLAN, PRI, IPv4 SIP, IPv6 SIP, IP IPv4 SIP, IPv6 DIP, Protocol Type, TCP source port, UDP source port, TCP destination port, and UDP destination port}	VFP-based {Max 2048}

a. The chip supports a total of 4096 SR flows, which can be a mix of seamless redundancy types.

NOTE: HSR and PRP packets with an unknown unicast MAC-DA are flooded the same as in a standard bridge. For IEEE 802.1CB, everything is pre-configured, no flooding happens, and unknown unicast MAC-DA packets are dropped.

6.3.5.1 HSR Network-Level Flow

HSR networks support ring topology. A source node sends duplicate packets to both directions of the HSR ring. The destination nodes accept one of the duplicate copies and discard the second copy.

6.3.5.1.1 Unicast Flow

Figure 57 shows an example of an HSR network with six nodes. In this figure, node 1 is the source node, and node 5 is the destination node for a unicast HSR flow.

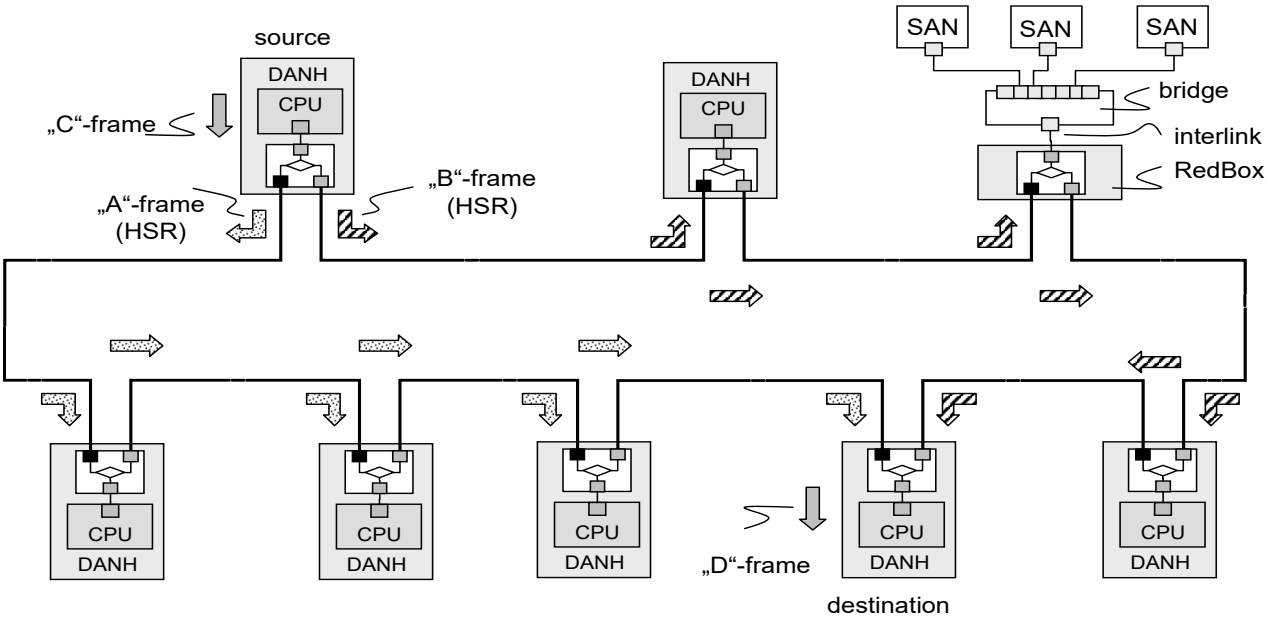
At the source node, the HSR Packet Replication function makes two copies, A and B, of the packet C. It then adds an HSR tag to both copies before transmitting them to different ports connected to the HSR ring. The sequence number in the HSR tag of copies A and B are the same, which enables the destination to identify the two frames, A and B, as copies. The NET_ID in copies A and B is different and indicates a clockwise or counter-clockwise copy.

Each intermediate node receives a copy from one ring port and forwards the copy to the other ring port, as illustrated in Figure 57.

Only the destination node receives both copies (A and B) on its ring ports. The Duplicate Discard Function in the destination node accepts one of the two copies and then removes the HSR tag to construct the unicast frame C. When the other copy is received, the Duplicate Discard Function rejects it as a duplicate after the sequence number in the HSR tag matches with the copy it already accepted.

If the unicast address has no known destination in the HSR ring, it will traverse the ring completely until it is received by the source node. The source node discards this message because the message was generated by the node itself. This function is accomplished by comparing the MAC-SA of the packet with the proxy MAC addresses stored in the node's MAC table.

Figure 57: Unicast Flow in an HSR Ring



6.3.5.1.2 Multicast Flow

Figure 58 shows an example of an HSR ring where a source node injects a duplicate multicast packet to the ring, and the multicast destination nodes each receive one copy of the multicast packet.

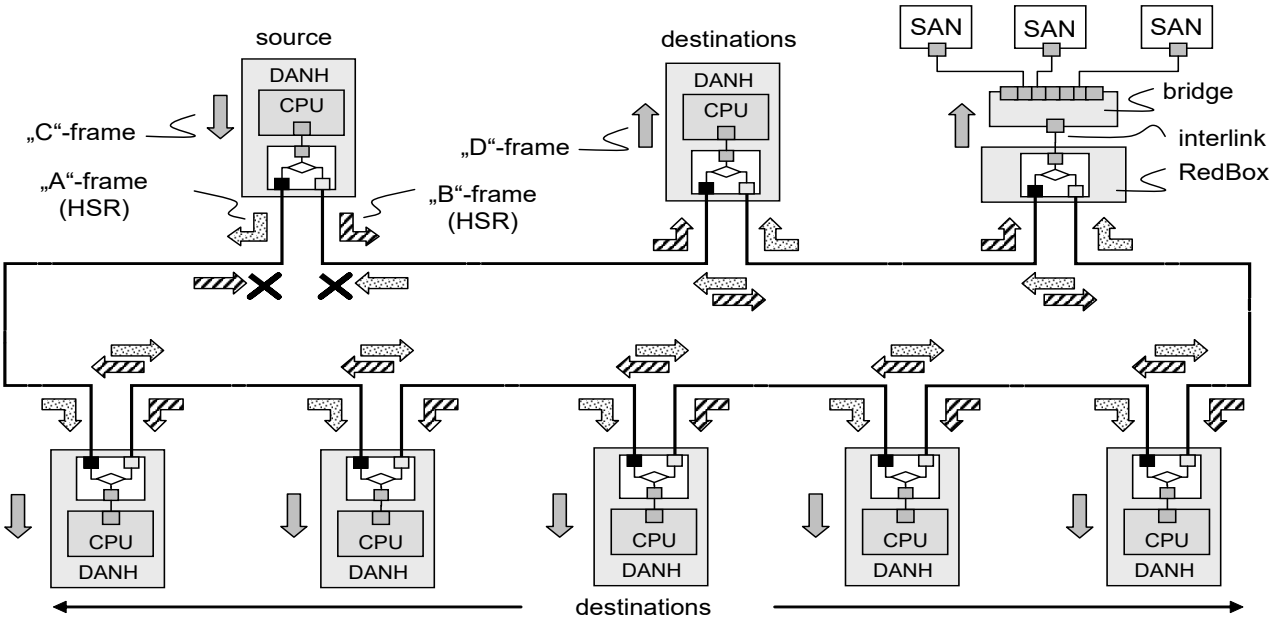
At the source node, the HSR Packet Replication function makes two copies, A and B, of the packet C. It then adds an HSR header to both copies before transmitting those to both ports connected to the HSR ring. The sequence number in copies A and B are the same, which enables a destination to identify the two frames (A and B) as copies of the same message. The LAN_ID is different in frames A and B, which allows the destination to differentiate the frames from each other.

Each destination node receives two copies (A and B) on the two ports connected to the HSR ring. The Duplicate Discard function in each destination node accepts one of the two copies and removes the HSR tag to construct multicast frame C. When the second copy of the same frame is received, the Duplicate Discard function rejects it as a duplicate after checking the sequence number in the HSR tag and determining that it matches the copy it already accepted.

In addition, each node must comply with the IEEE 802.1D bridge function. Except for the source node, all nodes also forward the multicast frame they receive on one ring port to the other ring port. As a result, the copies (A and B) of the multicast packet C travel in opposite directions of the ring at any intermediate node. However, when the source node receives the copies (A and B), it does not forward those received copies because it can identify itself as the source that injected the frames to the ring by comparing the MAC-SA of the message with the MAC-SA of proxy nodes stored in its MAC table.

NOTE: Optimization of the multicast flow is possible to prevent multicast packets from looping through the ring. This optimization can be achieved by ensuring that when a ring node received a frame with a specific sequence number, it does not forward the same frame if it receives it again on a different ring port.

Figure 58: Multicast Flow in an HSR Ring



6.3.5.2 PRP Network-Level Flow

PRP is similar to HSR; however, it uses two separate LANs to transport the duplicate packets to the destination. PRP can be considered as a subset of HSR, where there are only two nodes in the ring.

Figure 59 shows a PRP network model. As the figure shows, there are only two ring nodes. These nodes are the source and destination nodes, and LAN-A and LAN-B are transparent and behave like standard Ethernet switches with regard to PRP. However, a PRP node does not forward packets between its PRP ports.

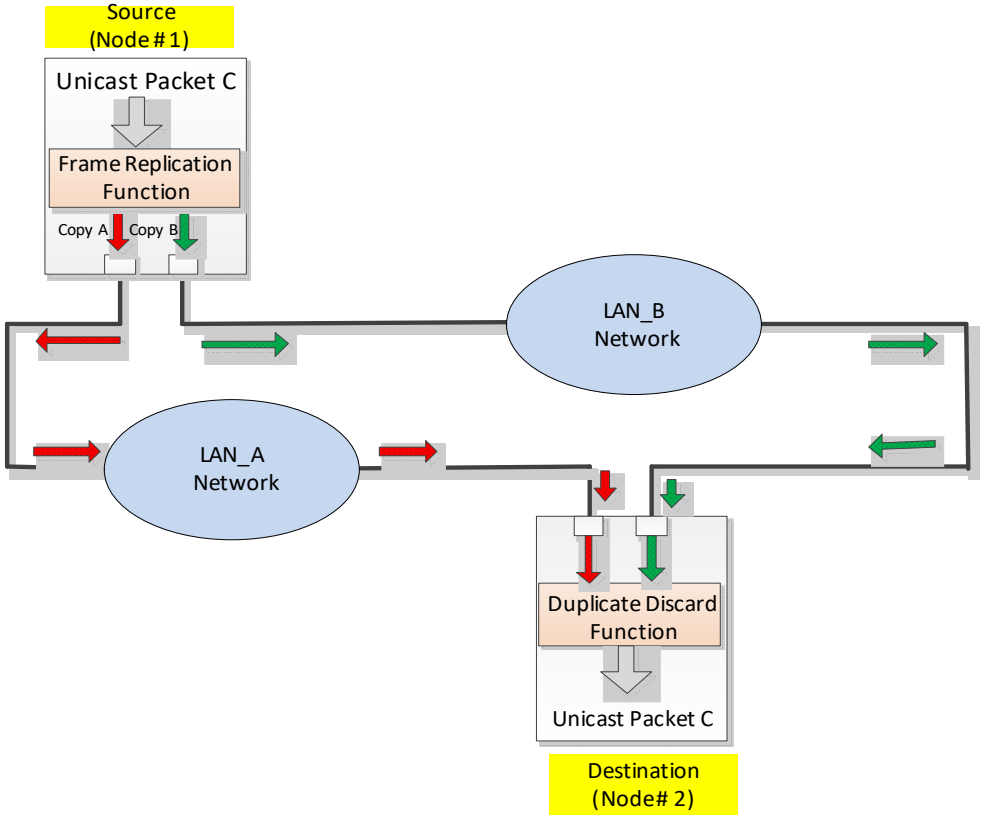
NOTE: The PRP Tag is different from the HSR tag. One main difference is that the PRP Tag is added to the trailer of the packet, while the HSR tag is added to the packet header.

The unicast and multicast PRP network flow is similar to HSR with the limitation that there is no transit-PRP node; instead, all transit nodes are standard Ethernet IEEE 802.1 bridges. Also, PRP nodes do not forward packets between their PRP ports, as opposed to HSR nodes that can forward packets between their HSR ports.

In Figure 59, the source node injects a packet to both LAN-A and LAN-B by adding a PRP Tag to the end of the packet. The PRP Tag for both duplicate packets will have the same sequence number but different LAN_IDs. LAN-A and LAN-B forward the packet normally. The destination node accepts the first copy that it receives and discards the second copy. If the MAC address is not known, the destination node does not forward the packet from LAN-A to LAN-B, or vice versa. Instead, it drops the packet.

NOTE: Unknown unicast flooding must be used in PRP.

Figure 59: PRP Network Model



6.3.5.2.1 SAN Forwarding

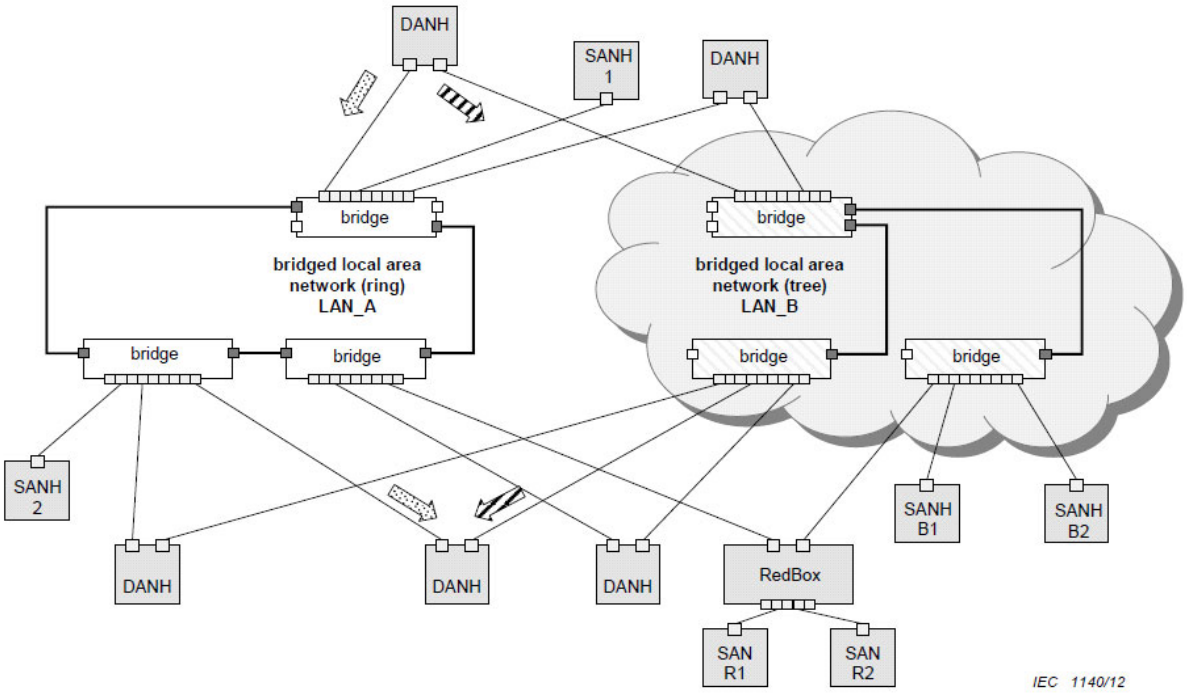
Figure 60 shows a scenario where a SAN is attached to LAN-A or LAN-B of a PRP network.

As the figure shows, a DAN box or Red-box forwards packets to other DAN boxes or Red-boxes by duplicating the packets to LAN-A and LAN-B with an RCT Tag, and the receiving DAN or Red-box performs a duplicate discard.

However, when a DAN or Red-box forwards packets to a SAN in LAN-A or LAN-B, it is not required to duplicate the packet, and it is not required to add a PRP RCT Tag. Also, when a SAN sends a packet to a DAN or to a Red-box, the packet does not have an RCT Tag.

The BCM56070 supports DANs and SANs in the PRP network, which means it can receive and transmit RCT Tagged and non-RCT Tagged packets on a PRP port.

Figure 60: SAN in PRP Network



6.3.5.3 PRP-HSR Interworking Network Flow

This section describes the network flow when one or more PRP networks are connected to an HSR ring. Theoretically, up to seven PRP networks can be connected to a single HSR ring.

Figure 61 shows a network-level flow from a PRP network to an HSR network. As the figure shows, an end node in the PRP network transmits two copies (A and B) of the same packet to LAN-A and LAN-B. Both copies (A and B) have a PRP Trailer with the same Sequence number but different LAN_IDs.

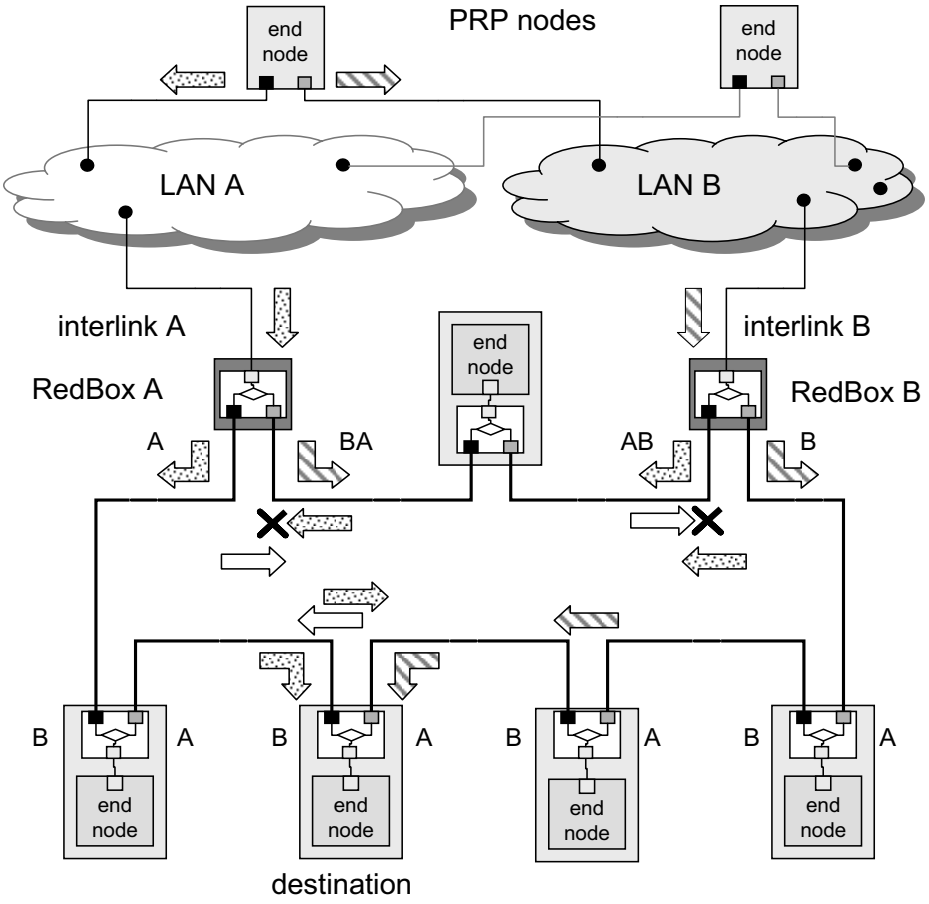
Packet A is received on the Interlink A on Red-box A, and packet B packet is received on the Interlink B on Red-box B. The same NET_ID is assigned to Red-box A and Red-box B, which indicates they are connected to the same PRP network.

Red-box A and Red-box B generate HSR duplicate packets and transmit them to both directions of the HSR ring. Red-box A and Red-box B translate the PRP Trailer into an HSR tag. The SR tag translation is performed as follows:

1. HSR Seq. Number = PRP Seq. Number
2. HSR NET_ID = 1
3. HSR LAN_ID = 0 for HSR packets sent counter-clockwise on the HSR ring
4. HSR LAN_ID = 1 for HSR packets sent clockwise on HSR ring

The destination node on the HSR ring accepts the first received packet (A or B) and discards the next duplicate packet.

Figure 61: PRP-HSR Internetworking



The following section describe the two types of behavior, which are configurable.

6.3.5.3.1 Option 1 (Default)

The duplicate packets generated by Red-box A toward Red-box B (packet BA), is discarded by Red-box B if Red-box B has already received packet B from its interlink. Similarly, duplicate packets generated by Red-box B toward Red-box A (packet AB), is discarded by Red-box A if Red-box A has already received packet A from its Interlink. The conditions for discarding AB by Red-box A and BA by Red-box B is that the packet's NET_ID should match the Red-box's NET_ID, and the packet's Sequence Number must have been previously seen on the Interlink. If these conditions are not met, the AB and BA packets are forwarded by Red-box A and Red-box B to the other ring port.

In option 1, if the MAC-DA address of A or B is not found on the HSR ring, or if the MAC-DA is multicast, packet A will be received by Red-box B, and packet B will be received by Red-box A. Red-box A and Red-box B will discard these packets because they have the same NET_ID as the NET_ID assigned to themselves, and the packet's Sequence Number has been seen previously on the Interlink.

6.3.5.3.2 Option 2

The duplicate packets generated by Red-box A toward Red-box B (packet BA) will be discarded by Red-box B because the packet's NET_ID matches Red-box B's NET_ID. Similarly, the duplicate packets generated by Red-box B toward Red-box A (packet AB) will be discarded by Red-box A because the packet's NET_ID matches the Red-box A's NET_ID.

In option 2, if the MAC-DA address of A or B is not found on the HSR ring, or if the MAC-DA is multicast, packet A will be received by Red-box B, and packet B will be received by Red-box A. Red-box A and Red-box B will discard these packets because the packet's NET_ID is the same as the NET_ID of the Red-Box.

6.3.5.4 HSR-PRP Interworking Network Flow

Figure 62: HSR-PRP Interworking

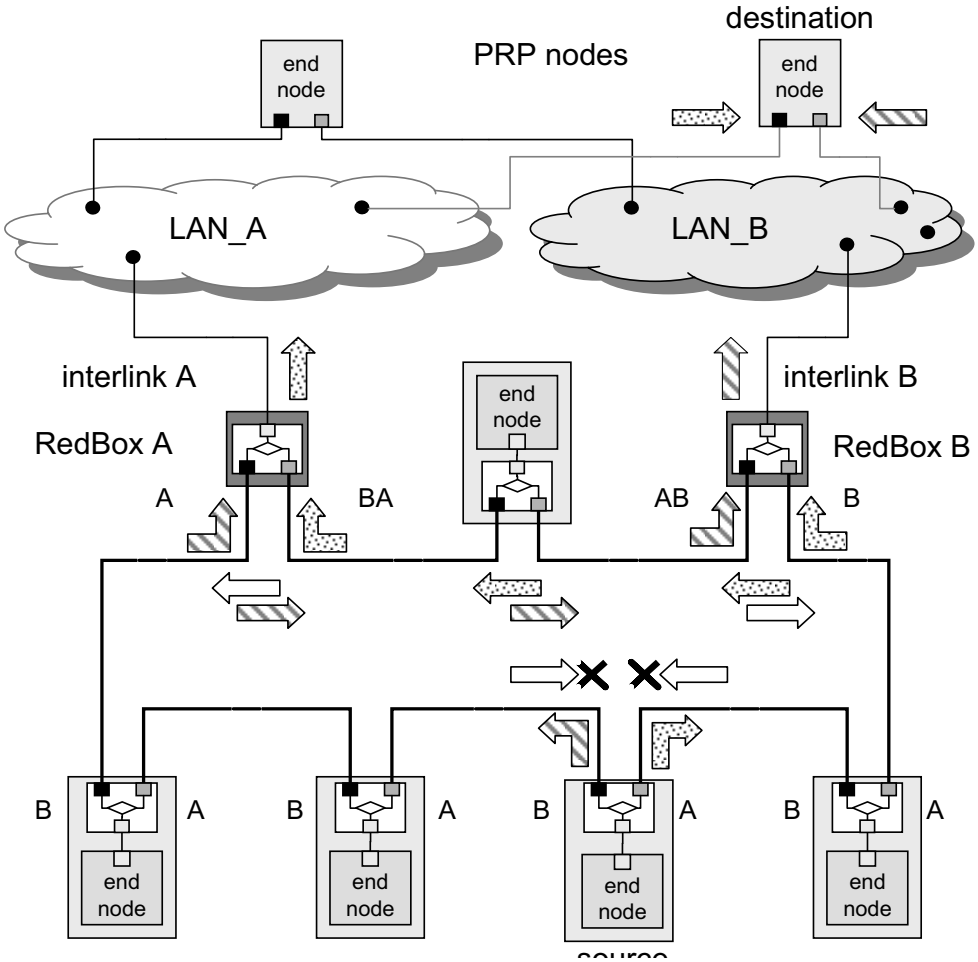


Figure 62 shows a network-level flow from the HSR network to the PRP network. As the figure shows, a source node in the HSR network transmits two copies (A and B) of the same packet to the HSR ring. Both copies (A and B) have an HSR tag with the same Sequence number and NET_ID = 0 (0 is reserved for packets originating in an HSR ring) but different LAN_ID.

When Red-box A receives packet A from the HSR ring, based on the MAC-DA, it forwards the packet to Interlink A. Similarly when Red-box B receives packet B from the HSR ring, based on the MAC-DA, it forwards the packet to Interlink B. Red-box A and Red-box B remove the HSR tag and add a PRP Trailer.

The Red-box A or Red-box B SR tag translation is as follows:

1. PRP Seq. Number = HSR Seq. Number
2. PRP LAN_ID = 0 for Red-Box A
3. PRP LAN_ID = 1 for Red-Box B

NOTE: For a multicast packet generated by an HSR node, in addition to forwarding the packet to Interlink A and B, Red-box A and Red-box B also forward the packet to their other ring port. Such multicast packets will eventually reach the source HSR node, where they will be discarded since their MAC-SA matches the proxy MAC-SA stored in the MAC table.

Figure 63: Multiple PRP Network to HSR Interworking

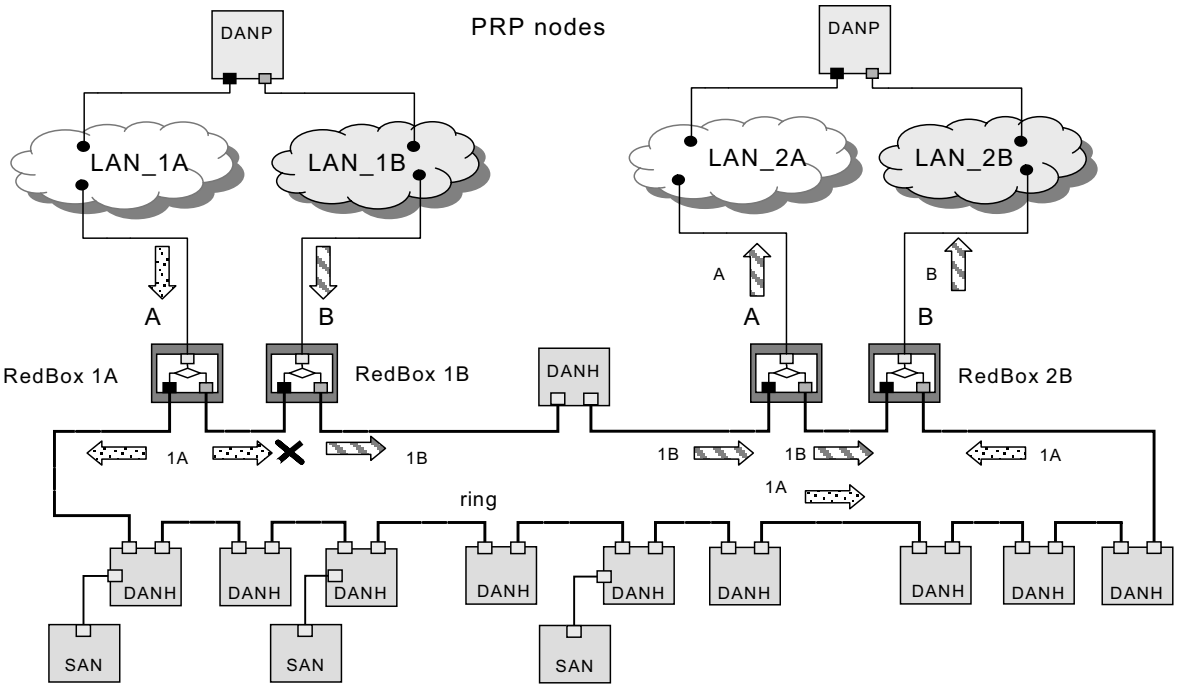


Figure 63 shows the case where two PRP networks (PRP Network 1 and 2) are connected to an HSR ring. Each PRP network connects to the HSR ring through two Red-boxes (Red-box A and Red-box B), where Red-box A is connected to PRP LAN-A, and Red-box B is connected to PRP LAN-B.

As the figure shows, an end node in PRP network 1 transmits two copies (A and B) of the same packet to LAN-1A and LAN-1B. Both copies (A and B) have a PRP Trailer with the same Sequence number but different LAN_IDs.

Packet A is received on the Interlink A of Red-box 1A, and packet B packet is received on the Interlink B of Red-box 1B. The same NET_ID=1 is assigned to Red-box 1A and 1B, indicating that they are connected to the same PRP network, PRP Network 1.

Red-box 1A and Red-box 1B generate HSR duplicate packets and transmit them to both directions of the HSR ring. Red-box 1A and Red-box1B translate the PRP Trailer to HSR tag. Red-box 1A or Red-box 1B Tag translation is performed as follows:

1. HSR Seq. Number =PRP Seq. Number
2. HSR NET_ID =1
3. HSR LAN_ID = 0 for HSR packets sent counter-clockwise on the HSR ring
4. HSR LAN_ID = 1 for PRP for HSR packets sent clockwise on the HSR ring

Red-box 2A and Red-box 2B have NET_ID= 2 assigned to them. When packet 1B is received by Red-box 2A, MAC-DA lookup is performed, and if the destination is in PRP network 2, the packet is forwarded to Interlink A of Red-box 2A because the packet NET_ID = 1 is not the same as Red-box 2A NET_ID = 2.

Similarly, when packet 1A is received by Red-box 2B, MAC-DA lookup is performed, and if the destination is in PRP network 2, the packet is forwarded to Interlink B of Red-box 2B because the packet's NET_ID = 1 is not the same as Red-box 2B NET_ID = 2.

Red-box 2A and Red-box 2B translate the HSR tag to a PRP Trailer. The Red-box 2A or Red-box 2B Tag translation is performed as follows:

1. PRP Seq. Number =HSR Seq. Number
2. PRP LAN_ID = 0 for PRP packets sent over Interlink A
3. HSR LAN_ID = 1 for PRP packets sent over Interlink B

6.3.5.5 HSR-HSR Ring Interconnect Network Flow

This section describes the network-level flow when two HSR rings are connected to each other through Quad-boxes. Figure 64 shows and example of this topology. The Quad-box forwards frames over each ring in the same way as any HSR node and passes the frames unchanged (with the same HSR tag) to the other ring unless the frame is identified as a frame that was already received from the other ring. There is no learning of MAC addresses in a Quad-box.

The presence of two Quad-boxes on the same ring causes two copies of the same frame to be transferred from the first ring to the second ring. Each Quad-box generates two copies, resulting in four copies of the same frame that exist on the second ring.

NOTE: The four copies do not circulate on the second ring because when a copy from the first Quad-box reaches the second Quad-box on the same second ring, the second Quad-box will not forward it if the Quad-box already sent a copy that came from its interlink. Conversely, if the second Quad-box did not yet receive a copy from its interlink, it will forward the frame but not the copy that comes later from the interlink.

NOTE: For HSR-HSR interworking all packets are treated as multicast. When a Quad-box receives a frame, it forwards the frame to its other ports if it did not already send a copy.

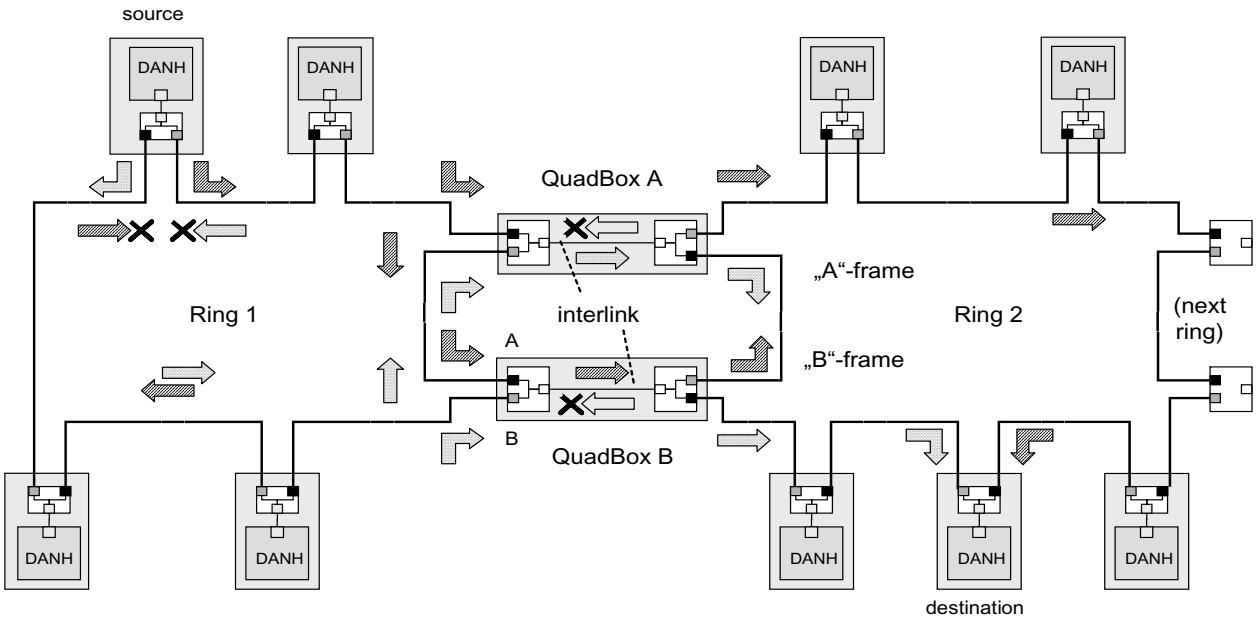
In Figure 64, a source node in HSR ring 1 generates two copies (A and B) of the same packet. Copy A is sent counter-clockwise on Ring 1, and copy B is sent clockwise on Ring 2. When copy A is received by Quad-box A, it replicates the packet to its other three ports, of which two ports are in HSR ring 2, and the third port is in HSR ring 1.

Quad-box B will receive both copies (A and B) from HSR ring 1. Packet A and B are forwarded to the other HSR ring 1 port of Quad-box B. However, only copy A or B, whichever is received first, is forwarded to both of the HSR ring 2 ports of Quad-box B, and the other copy is not forward to HSR ring 2 at all.

When Quad-box A receives packet B from Quad-box B on HSR ring 2, it discards the packet if it has already received a similar packet with the same sequence number from HSR ring 1. Otherwise, Quad-Box A will forward packet B to its other three ports. Similar behavior is applicable to Quad-box B.

The source node in the HSR ring will eventually receive copies A and B of the packet it generated. In that case, the source node discards the packets because the MAC-SA is in its proxy MAC table.

Figure 64: HSR-HSR Ring Interconnect Network Flow



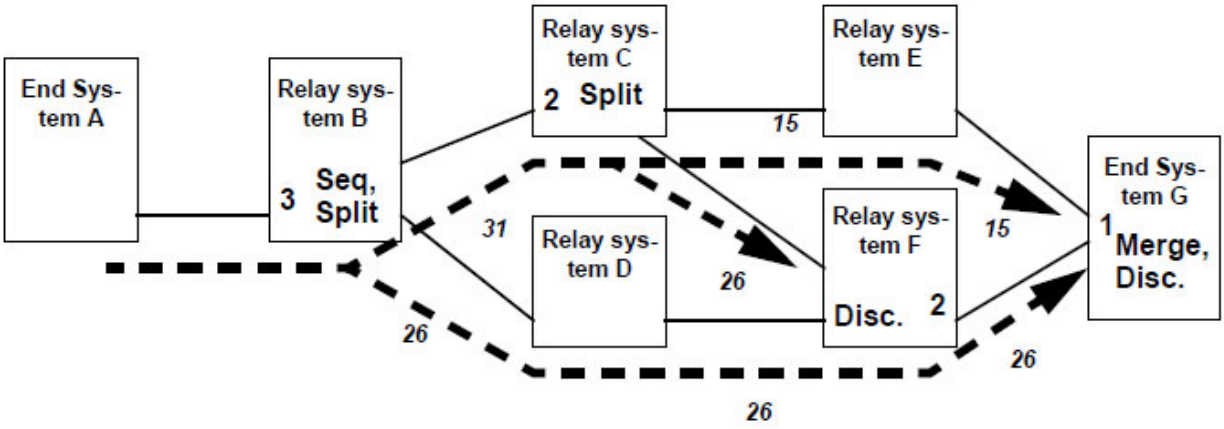
6.3.5.6 HSR Mesh Network

An HSR network can be a mesh network as shown in Figure 65.

As the figure shows, node B is a Red-box that acts a proxy for node A and generates duplicate HSR packets. Node G is a Red-box and performs duplicate discard and sends one of the duplicate packets to its interlink.

However, nodes C, D, E and F act as *relay* nodes. These nodes receive and transmit HSR-tagged packets only. For example, node C receives an HSR-tagged packet and duplicates that packet to nodes E and F without removing or changing the HSR tag. Node F receives HSR packets from nodes C and D, performs a duplicate discard, and sends only one copy to node G, but it does not remove or change the HSR tag.

Figure 65: HSR Mesh Network



6.3.5.7 IEEE 802.1CB Network-Level Flow

IEEE 802.1CB is very similar to HSR and supports ring, star and mesh topologies. The only difference is that IEEE 802.1CB flows are based on finer-grained flows, such as flows based on (DA, VLAN, PRI) or (DA, VLAN, PRI, SIP, DIP, Protocol, S-Port, D-Port).

Another difference between IEEE 802.1CB and HSR is that the LAN_ID is encoded as a VLAN in IEEE 802.1CB. IEEE 802.1CB flows can coexist with non-seamless redundant flows, where normal Ethernet flows use a different VLAN.

6.3.6 System and Node Level Flows

This section describes system and node level flows for each of the HSR, PRP, and IEEE 802.1CB.

NOTE: This document assumes a single-chip system and does not support multichip system or HiGig. Therefore, in this document, a system and node are considered the same.

This section describes all possible unicast and multicast flows for HSR, PRP, and IEEE 802.1CB nodes.

6.3.6.1 Switch Model

In this section, the BCM56070 is Red-box that acts as proxy DAN node for devices (sensors, actuators, and so on.) attached to it that do not have seamless redundancy capability. The switch may also act as Quad-box connecting two HSR rings to each other.

Figure 66 shows the switch modes. As the figure shows, the switch operates as an Ethernet bridge, which has a normal non-SR-side (standard Ethernet side) and an SR-side. The SR-side has an added functionality called *LRE*. LRE is the entity that duplicates a packet received from the non-SR-side to two ports (A and B). The LRE discards duplicate packets received from the SR-side that are destined to the non-SR -side.

Figure 66: Bridging Model of an SR Switch

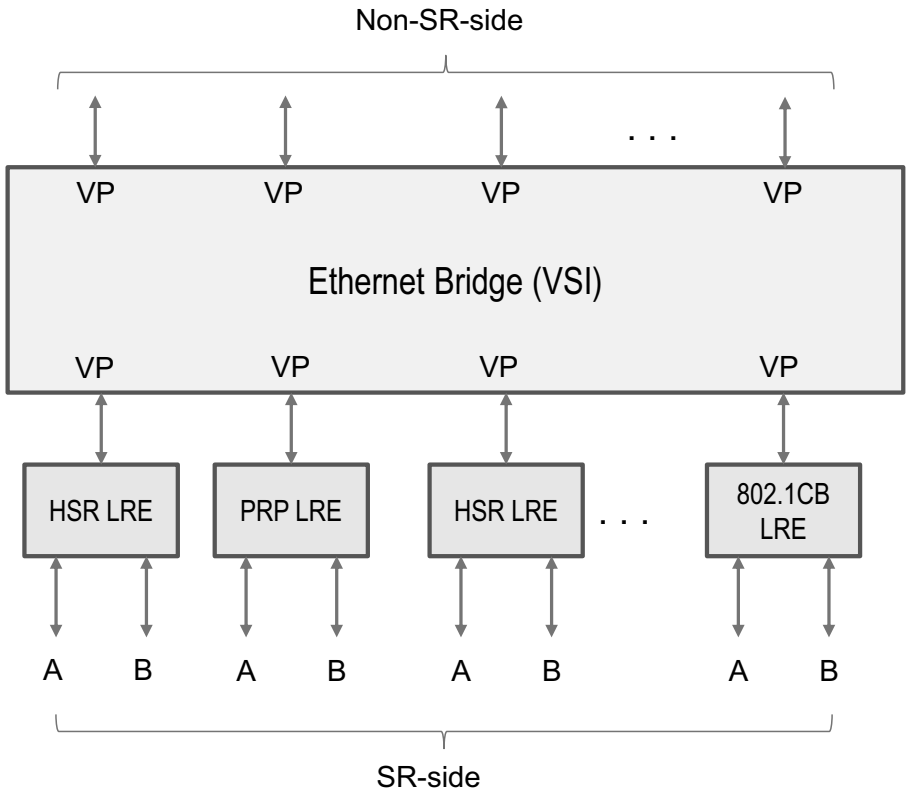
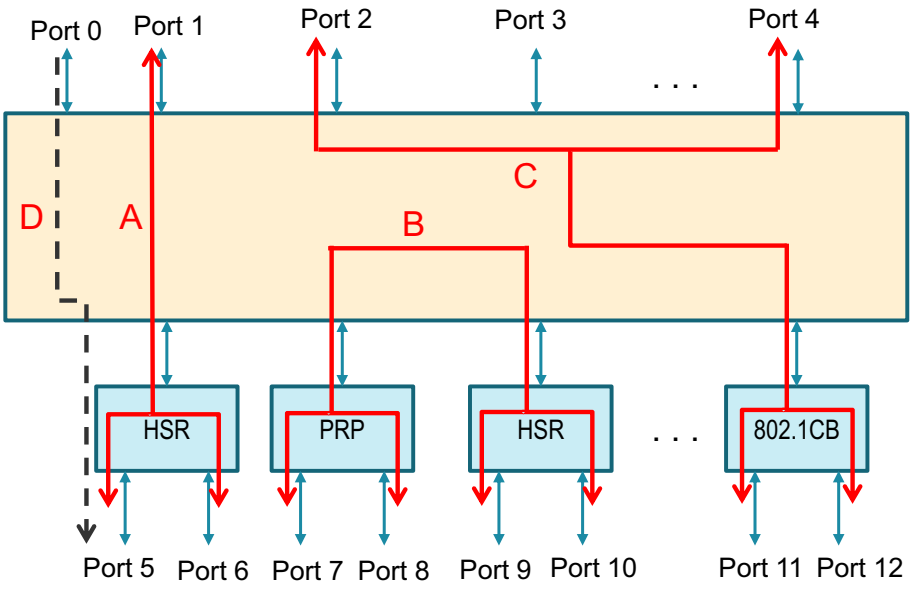


Figure 67 shows four flow examples (flow A, B, C, and D). Flow A is an HSR flow between ports 1, 5, and 6. Port 1 is a non-HSR-port, and ports 5 and 6 are HSR-ports. In flow A, the switch will add an HSR tag to packets forwarded to port 5 and 6 and will remove the HSR tag from packets received from ports 5 and 6 and forwarded to port 1.

Flow B is an interworking flow (Quad-box) between two HSR ports (port 9 and port10) and two PRP ports (port 7 and port 8).

Flow C is an example of an 802.1CB flow. Ports 2, 3, and 4 are Interlink ports, and ports 11 and 12 are the 802.1CB ports. Frames of an 802.1CB flow may come from different Interlinks because the flow is identified based on (DA, VLAN, PRI).

Figure 67: SR-Flow Examples in BCM56070



6.3.6.2 Filtering

A Seamless redundancy mode supports the following three types of frame filtering:

- NET_ID filtering
- MAC-SA filtering
- Duplicate filtering

NET_ID filtering and discard is applied when an HSR frame with a NET_ID is received by a Red-box and needs to be forwarded to a PRP Interlink port that is attached to a PRP network and has the same NET_ID as the frame's NET_ID. In this case, the frame must not be forwarded.

MAC-SA filtering and discard is applied when an HSR, PRP, or 1CB frame is received by a Red-box and the MAC-SA of that frame is a proxy MAC address. A MAC address is a proxy MAC address when the Red-box generates a duplicate copy and filters duplicate copies for frames with that MAC address as a MAC-SA.

Duplicate filtering and discard is applied when an HSR, PRP, or 1CB frame is received and the sequence number check indicates it is a duplicate frame or a late frame and is therefore discarded.

6.3.6.3 PRP Node (Red-Box)

PRP is the simplest redundancy protocol. A PRP node forwards traffic only between a non-PRP port and the PRP ports. Unlike HSR, in PRP, no traffic forwarding is done between PRP ports.

A PRP node (DANP) has the following three modes:

- Duplicate discard
- Duplicate Discard with Transparent reception
- Duplicate accept

The SDK should configure the forwarding tables and present the three modes for users to select from.

In PRP, forwarding between PRP ports is not allowed. Also, forwarding for unicast and multicast is the same in PRP nodes, meaning that even multicast packets are not forwarded between PRP ports.

The two types of PRP nodes are as follows:

- A PRP end node (DANP) has three ports that include two LAN ports and a Link Layer port that connects to the CPU.
- A PRP Red-box has three ports that include two LAN ports and a Interlink port that connects to a SAN. A PRP Red-box can also include a Link Layer port for connection to a CPU.

6.3.6.4 PRP Duplicate Discard Mode (Default Mode)

PRP duplicate discard mode is the default mode of operation. A sender in PRP duplicate discard mode duplicates a packet received form its Interlink or Link Layer to the PRP ports and adds a Redundancy Control Tag (RCT) to the end of the packet that encodes the same sequence number in both duplicate packets.

In PRP duplicate discard mode, a receiver forwards the first copy of the duplicate packet to the upper layer or Interlink and discards the second copy.

Figure 68: PRP Node (Duplicate Discard Mode)

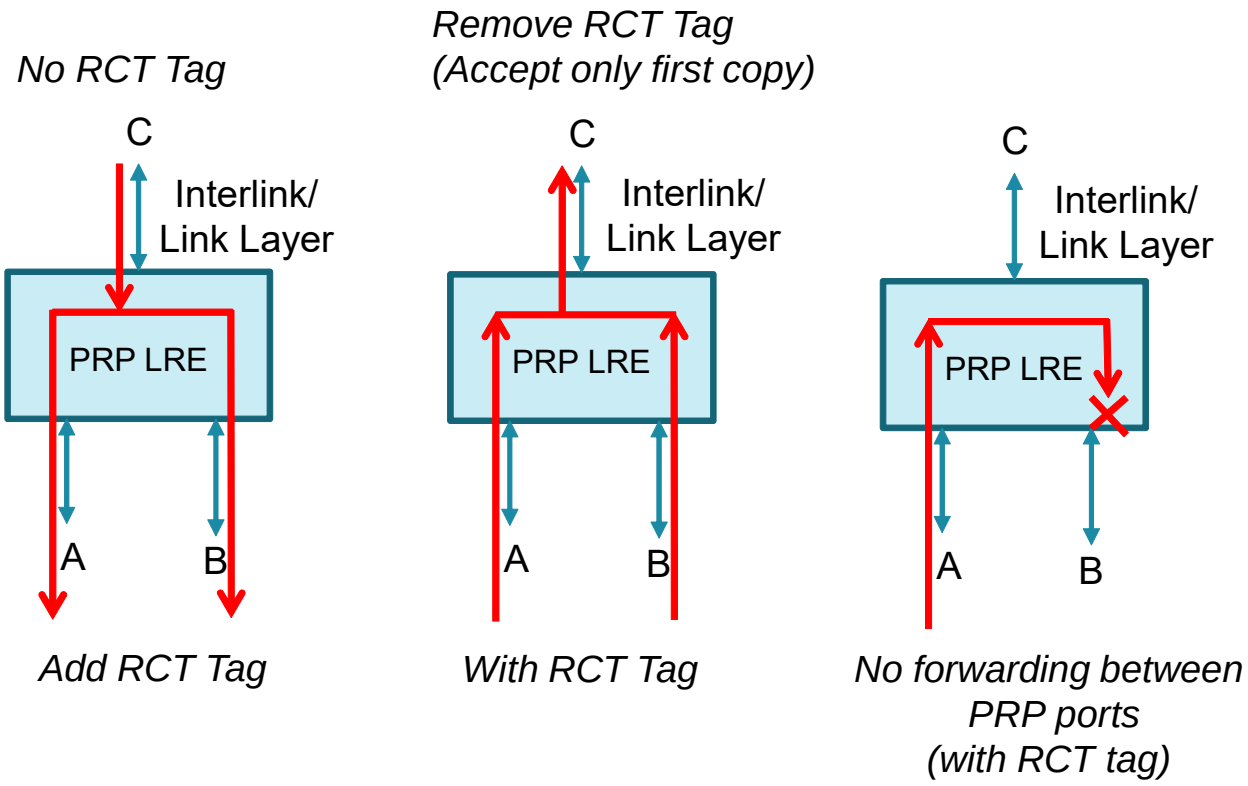


Figure 68 shows the node behavior under duplicate discard mode.

In the TX direction forwarding behavior (Interlink or Link layer → PRP ports), the following logic applies:

- If the destination MAC address belongs to the reserved addresses in IEEE 802.1Q, treat the frame at the link layer.
- Else, if the destination MAC address belongs to a SAN Forward the packet as per IEEE 802.1Q.
- Else, if the destination MAC address belongs to DAN, do the following:
  - Duplicate the packet to both PRP LAN-A and LAN-B ports,
  - Add 6 byte RCT Tag to each packet.
  - Increment last transmitted sequence number by 1.
  - Copy the sequence number to RCT Tag.
  - Set LAN_ID = 0 for copy sent to LAN-A port.
  - Set LAN_ID = 1 for copy sent to LAN-B port.
  - Increment the TX counter for LAN-A and LAN-B ports.
- End.

In the RX direction forwarding behavior (PRP port → Interlink or Link layer), the following logic applies:

- If the frame is erroneous, increment the error counter of the respective LAN or port and drop the frame.
- Else, if the destination MAC address belongs to the reserved addresses in IEEE 802.1Q, treat the frame at the link layer.
- Else, if the frame is a duplicate, do the following:
  - Increment RX Counter for the received LAN port.
  - Discard it.
- Else:
  - Increment RX Counter for the received LAN port.
  - Remove the RCT Tag and forward the frame to the upper layers.
- End

The duplicate discard method forgets an entry after the entry age out time expires.

6.3.6.5 HSR Node (Red-Box)

HSR is another standard for seamless redundancy. In addition to the mesh and star topologies, HSR supports the ring topology and can forward ring traffic between its redundant ports, which makes it more complex than PRP. In addition, the HSR nodes supports many modes of operation in addition to the PRP modes.

An HSR Node (DANH) is operable in one of the following modes, which are changeable at runtime using management commands:

- Mode H (default mode): HSR-tagged forwarding – In this mode, the DANH inserts the HSR tag on behalf of its host and forwards the ring traffic. Duplicate frames and frames where the node is the unicast destination of the frame are not forwarded.
- Mode N (optional): No forwarding – In this mode, the node behaves like mode H with the exception that it does not forward ring traffic from port to port. This mode is similar to PRP duplicate discard mode.
- Mode T (optional): Transparent forwarding – In this mode, the DANH removes the HSR tag before forwarding the frame to the other port or to the Interlink or Link Layer. Also, in this mode, when forwarding a frame from the Interlink or Link Layer to other ports, an HSR tag is not added. Additionally, in this mode, duplicate frames are not discarded.
- Mode M (optional): Mixed forwarding HSR-tagged and non HSR-tagged – In this mode, the DANH inserts the HSR tag depending on local criteria when injecting frames into the ring. HSR tagged frames from ring ports are handled according to Mode H. Non-HSR tagged frames from ring ports are handled according to IEEE 802.1D forwarding rules.

- Mode U (optional): Unicast Forwarding – In this mode, the node behaves as in Mode H, except that it forwards unicast traffic for which it is the destination as it does for multicast.
- Relay Mode – In this mode, the node does not add or remove the HSR tag. When it receives an HSR frame from the Interlink, it duplicates the packet to both HSR ports and keeps the HSR tag. When the node receives HSR frame from one of the HSR ports, it performs duplicate discard and forwards the packet to the other HSR port or to the Interlink and keeps the HSR tag.

The two types of HSR nodes are as follows:

- An HSR end node (DANH) has three ports that include two ring ports and a Link Layer port that connects to the CPU.
- An HSR Red-box has three ports that include two ring ports and an Interlink port that connects to a SAN, a bridge, or another network. An HSR Red-box can also include a Link Layer port for a connection to the CPU.

NOTE: Unlike PRP nodes, unicast and multicast forwarding is not the same in HSR nodes.

6.3.6.6 HSR Mode H (Default Mode)

This is the default mode of operation of HSR nodes. In this mode, an HSR node duplicates the packets it receives from its Interlink or Link layer to its HSR ports and adds 6 bytes of HSR to the packet. In this mode, an HSR node accepts the first copy of an HSR packet from its HSR port and discards duplicate copies of the same HSR packet.

Unlike PRP nodes, an HSR node can also forward packets between its HSR ports.

Figure 69: HSR Node (Mode H, Unicast)

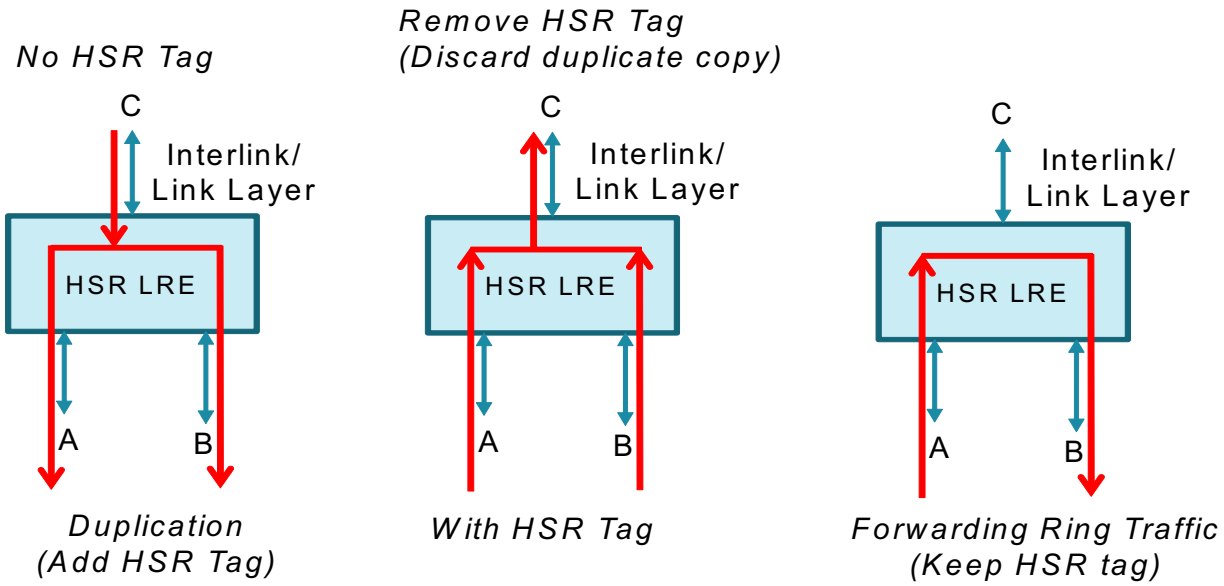


Figure 69 shows the unicast forwarding behavior of an HSR node in Mode H. This forwarding is very similar to PRP duplicate discard mode, with the exception that forwarding between HSR ports is allowed in HSR nodes.

When a packet is received by an HSR port, a MAC-DA lookup is performed, and if the MAC-DA indicates the destination is Interlink or Link layer, then the first copy of that packet is forwarded to the Interlink or Link layer, and further copies of the same packet are discarded. However, if the MAC-DA lookup indicates the destination is not an Interlink or Link layer, but it is in the HSR ring, the packet is forwarded to other HSR port.

In some cases, the MAC-DA of destinations on the ring may not be configured in the L2 table of transit nodes on the ring. In this type of case, the MAC-DA lookup fails, and the default is to flood the packet to the other ring port.

In some cases, a unicast packet that is injected to HSR ring by a source HSR node may not reach its destination and may be received back by the source HSR node itself. Such packets must be discarded by the HSR node based on the MAC-SA lookup, which indicates that the HSR node is the proxy node for that MAC-SA, and the HSR node was the node generating that HSR packet.

Frames with a destination MAC address according to IEEE 802.1Q 2004 Table 7-10 are treated at the Link Layer and are not expected to carry an HSR tag, even if it is sent by a DAN.

The duplicate discard method forgets an entry after the entry age-out time.

Figure 70: HSR Node (Mode H, Multicast)

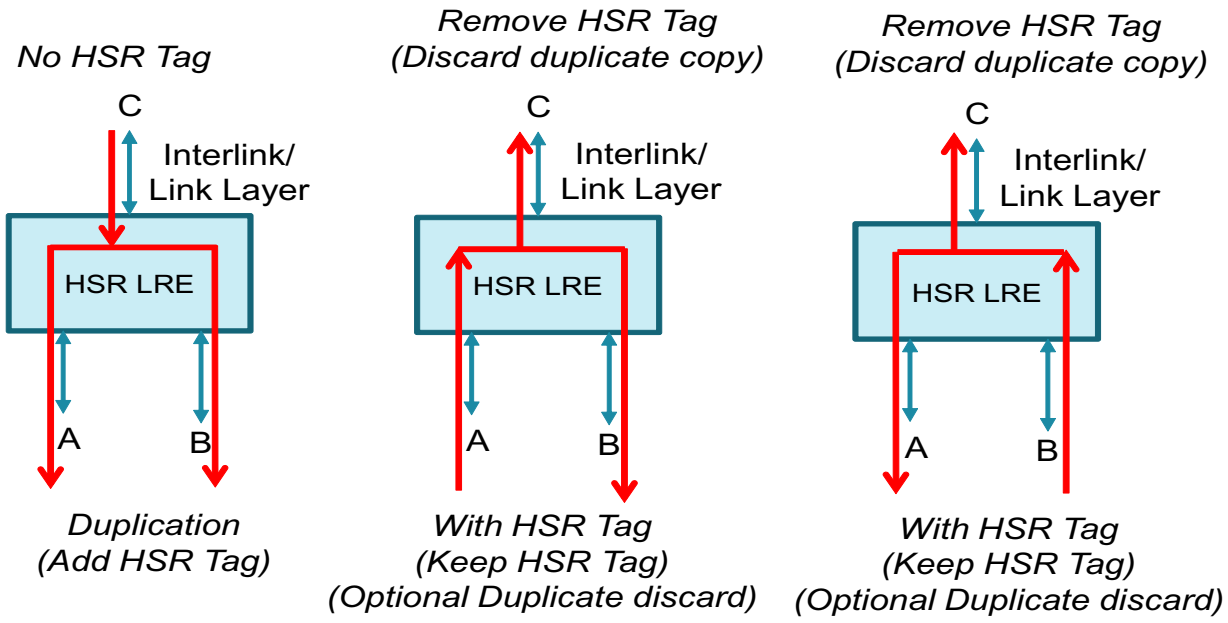


Figure 70 shows HSR Node H behavior when a multicast packet is received. As the figure shows, a multicast packet received from an Interlink or Link layer will be duplicated to both HSR ports with an HSR tag the same as for unicast packets.

However, when the multicast packet is received from an HSR port, a copy is sent to the Interlink or Link layer without the HSR tag (the HSR tag is removed), and another copy is sent to the other HSR port with the HSR tag (the HSR tag is kept).

When a multicast packet is received from an HSR port, duplicate discard logic is applied for the copy sent to the Interlink or Link layer. However, for the copy sent to the other HSR ports, the discard duplicate can be optionally applied.

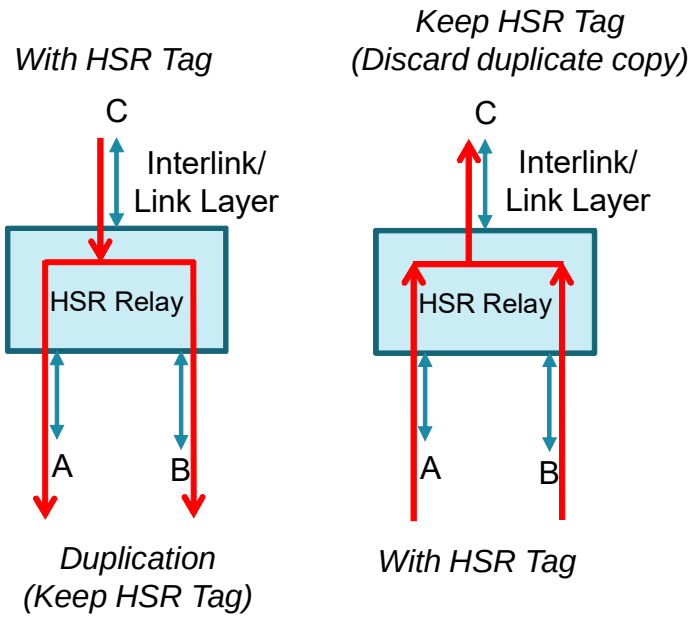
NOTE: When a multicast packet is injected into the HSR ring by a source HSR node and not discarded by any HSR node, it may be received back by the source HSR node itself. Such packets must be discarded by the HSR node based on the MAC-SA lookup, which indicates that the HSR node is the proxy node for that MAC-SA, and the HSR node was the node generating that HSR packet.

6.3.6.7 HSR Relay Mode

In some HSR topologies, such as a mesh topology, an HSR node may behave as an HSR relay node. All ports on an HSR relay are HSR ports and must use an HSR tag. Figure 71 shows an example of HSR relay nodes.

Figure 71 shows the behavior of an HSR node in relay mode for unicast or multicast forwarding. As the figure shows, when an HSR relay node receives a packet from its Interlink port (port C), which is already HSR tagged, it duplicates that packet to port A and B and keeps the HSR tag unchanged. In the other direction, when a relay node receives HSR tagged packets from port A or B, it performs a duplicate discard and forwards the unmodified packet to port C (Interlink) and keeps the HSR tag unchanged.

Figure 71: HSR Node Relay Mode Unicast or Multicast



6.3.6.8 HSR-HSR Quad-Box

Two HSR rings can be connected to each other using a Quad-box, which has four HSR ports. The behavior of a Quad-box is describes in this section. Quad-boxes perform only multicast forwarding. A Quad-box forwards a packet received on any of its four ports to the other three ports if it has not already received that packet on any of the other three ports.

Figure 72: HSR Quad Box (Multicast)

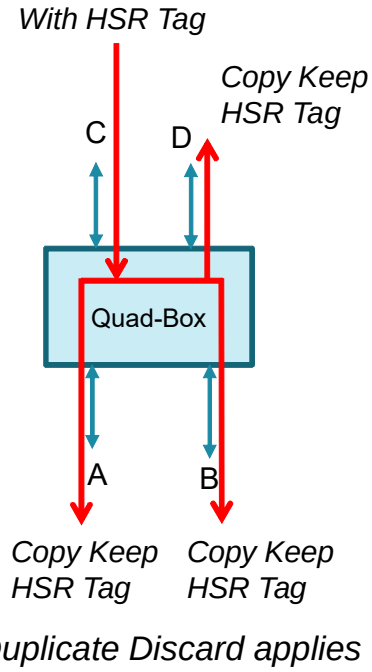


Figure 72 shows the node model of an HSR Quad-box. In this figure, when Port C receives an HSR packet, the Quad-box checks whether it has received the same packet from any of its other ports. If the Quad-box has received the packet, then it will discard it as duplicate; otherwise, it replicates the packet to the other three ports (A, B, and D) without modification and keeps the HSR tag unchanged. The same behavior applies when the packet is received from any of the other ports.

6.3.6.9 PRP-HSR Interworking Node

As described in previous sections and shown in Figure 48, PRP and HSR Interworking, Figure 62, HSR-PRP Interworking, and Figure 63, Multiple PRP Network to HSR Interworking, the Interlink of an HSR node can connect to a PRP LAN.

This section describes the expected behavior of this type of PRP-HSR node. One distinction of the PRP-HSR node is that the PRP and HSR tags must be translated when going from PRP to HSR, and vice versa.

Figure 73: HSR-PRP Interworking Node (Unicast)

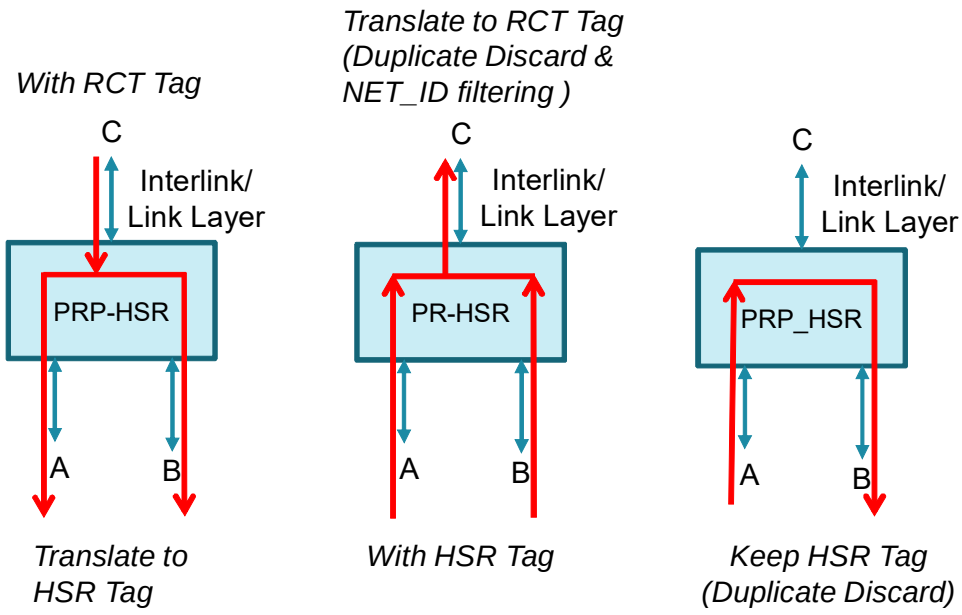


Figure 73 shows the unicast forwarding behavior of an HSR-PRP interworking node. As this figure shows, when a packet is received from a PRP network over the Interlink, it is duplicated to both HSR ports, and the PRP Trailer (RCT) is translated into an HSR tag.

When a packet is received from an HSR port and the destination is the Interlink, the packet is discarded if it is a duplicate packet; otherwise, the packet is forwarded to Interlink, and the HSR tag is translated into an PRP Trailer (RCT).

When a packet is received from an HSR port and the destination is an HSR network, the packet is discarded if it is a duplicate packet; otherwise, the packet is forwarded to the other HSR port, and the HSR tag is left unchanged.

Figure 74: HSR-PRP Interworking Node (Multicast)

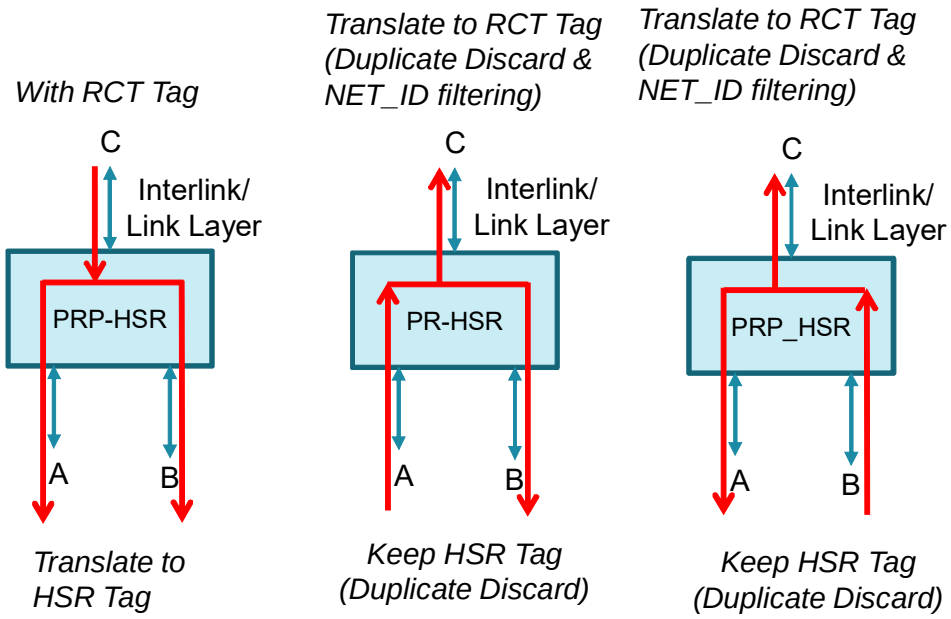


Figure 74 shows the multicast forwarding behavior of an HSR-PRP interworking node. As the figure shows, when a multicast packet is received from a PRP network over the Interlink, it is duplicated to both HSR ports, and the PRP Trailer (RCT) is translated into an HSR tag.

When a multicast packet is received from an HSR port, the packet is discarded if it is a duplicate packet, regardless of whether the packet was received originally from Interlink or any HSR port. If the packet has a NET_ID that is the same NET_ID assigned to the Interlink, then the packet is only forwarded to the other HSR port, and the HSR tag is preserved. Otherwise, if the packet has a NET_ID that is different from the NET_ID assigned to the Interlink, the packet is forwarded to the Interlink and the other HSR port. The HSR tag for the copy forwarded to the Interlink is removed and replaced with an RCT Tag, and the copy forwarded to the other HSR port leaves the HSR tag unchanged.

6.3.7 Chip-Level Packet Flow

This section describes the chip-level flow for seamless redundancy.

NOTE: Only single-chip operation is supported and assumed for seamless redundancy.

6.3.7.1 Node and Proxy Tables

HSR, PRP, and IEEE 802.1CB require the presence of a node table and a proxy table for seamless redundancy. The node table stores the MAC address of remote nodes that send duplicate packets to this node, and the proxy table stores the MAC address of the nodes that this node proxies and generates duplicate packets on its behalf.

The BCM56070 uses the L2_ENTRY table as both the node table and the proxy table. Acting as a node table, the L2_ENTRY table stores the MAC addresses of remote nodes sending packets to this node through PRP, HSR, or 1CB ports. Acting as a proxy table, the L2_ENTRY table stores the MAC addresses of the nodes sending packets to this node through Interlink or Link Layer ports.

NOTE: The BCM56070 requires the MAC-SA learning to be done in software rather than hardware. This is because, as part of the MAC learning, the new MAC address must be added to the L2_ENTRY table, the Flow_ID must be assigned, and the sequence number generation and tracking tables must be initialized.

6.3.7.2 Sequence Number RX Processing

Seamless redundancy requires that an SR node generate duplicate packets and add an incrementally increasing sequence number to the SR tag. In the RX direction, seamless redundancy requires an SR node to accept the first copy of the packet and discard the duplicate copy.

The BCM56070 does not reorder packets that are received out of order. The reordering function is left to the final receiver (sensor or actuator).

Figure 75: Sequence Number Regions

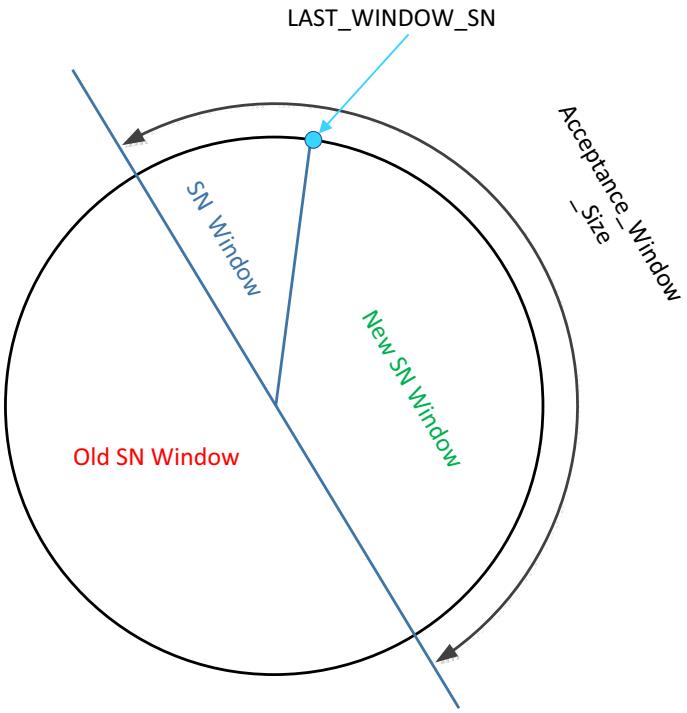


Figure 75 shows the sequence number regions. As the figure shows, the SN_Window region (marked in blue) has a size of SN_Window_Size. This is the current tracking window, and received sequence numbers are mostly expected to fall within this window. The duplicate discard algorithm is applied to this region because it accepts the first copy and discards duplicate copies.

The Acceptance_Window starts from the first sequence number in the SN_Window and has a size of Acceptance_Window_Size. Sequence numbers that fall in the acceptance window and are larger than the LAST_WINDOW_SN are considered to be new sequence numbers (green). Received frames in this region are normally accepted unconditionally and move the SN_WINDOW so that the LAST_WINDOW_SN = PKT.SN.

The OLD_Window (indicated in Red) is the region between the SN_Window and the Acceptance_Window. Packets that fall within this region may be accepted or discarded, based on an IFP rule. Frames in this region do not move the SN_WINDOW.

Figure 76: Sequence Number Window Table

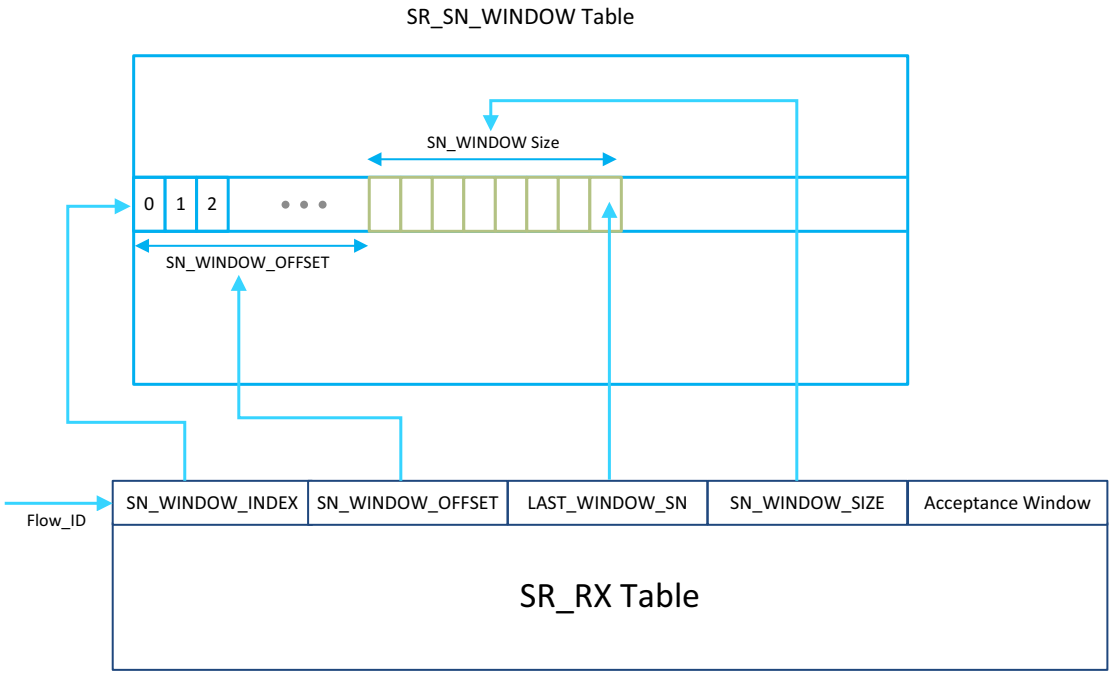


Figure 76 shows how the sequence number window is configured in the chip. The FLOW_ID is used to index the SR_RX table. The SR_RX table has the SN_WINDOW_INDEX to the SR_SN_WINDOW table. The SR_RX also provides the SN_WINDOW_SIZE, SN_WINDOW_OFFSET, and LAST_WINDOW_SN.

The sequence number window is configured in the SR_SN_WINDOW table. The SN_WINDOW starts from the SN_WINDOW_OFFSET and has a size equal to SN_WINDOW_SIZE.

Each bit in the SN_WINDOW represents a sequence number. The right-most bit of the SN-WINDOW corresponds to the LAST_WINDOW_SN that is stored per-flow in the chip. The left-most bit in the SN_WINDOW corresponds to (LAST_WINDOW_S – SN_WINDOW_SIZE +1).

For each flow, the SN_WINDOW is reset using one of the following three methods:

- Hardware reset, method one:
  - LAST_WINDOW_SN = 0
  - SN_WINDOW = '000000…00'
  - SN_RESET_STATE =1 //ACDPETANCE_WINDOW_SIZE = 2^{16}

    Always Accepts the first packet after reset and shift the SN_WINDOW to the new Sequence number, then set:
  - SN_RESET_STATE=0
  - Use SR_RX.ACCEPTANCE_WINDOW
  - LAST_WINDOW_SN = PKT.SN
  - Apply the normal duplicate discard algorithm
- Hardware reset, method two:
  - LAST_WINDOW_SN = 65535
  - SN_WINDOW = 111111…10
  - ACCEPTANCE_WINDOW = SR_RX.ACCEPTANCE_WINDOW
  - Apply the normal duplicate drop algorithm
- Software reset

In hardware reset, method one, the first packet after the reset is always accepted and shifts the SN_WINDOW by setting the LAST_WINDOW_SN = PKT.SN. Immediately after reception of the first packet, the hardware then sets SR_RX.ACCEPTANCE_WINDOW = ACCEPTANCE_WINDOW, and after this point, the normal duplicate discard algorithm is applied.

In hardware reset method two, the acceptance of the first packet after the SN Window reset follows the duplicate discard algorithm as usual.

In the software reset method, the CPU resets the SN Window. In this method, the SN_WINDOW aging is reported to the CPU.

When a sequence number is received and is within the SN_WINDOW, and the corresponding sequence number bit is 0, the packet is not a duplicate packet, and the corresponding bit is set to 1. When a packet is received and the corresponding sequence number bit is 1, then the packet is a duplicate. The IFP will decide to forward or discard duplicate or non-duplicate packets.

When a packet is received and the sequence number is before the LAST_WINDOW_SN – SN_WINDOW_SIZE +1, it is in the old sequence number region. The IFP can decide to accept or discard such packets.

When a packet is received and the sequence number is after the LAST_WINDOW_SN (in the green area in Figure 76), the whole SN_WINDOW is shifted to the new sequence number. Again, the IFP can decide to forward or discard such packets.

NOTE: The SN_WINDOW shifting happens in hardware regardless of whether the packet is accepted by the IFP.

6.3.7.3 Sequence Number TX Setting

When packets are received from the Interlink or Link Layer and duplicated to HSR, PRP, and 1CB ports, the sequence number starts from zero and increments by 1 for each packet duplicated. The sequence number is incremented only when the packet belongs to a valid SR flow (Flow_ID != 0). The BCM56070 supports only 16-bit sequence numbers, which are stored in the SR_TX table.

NOTE: The sequence number field in IEEE 802.1CB is 28 bits, but only the 16 LSB bits are used in the BCM56070.

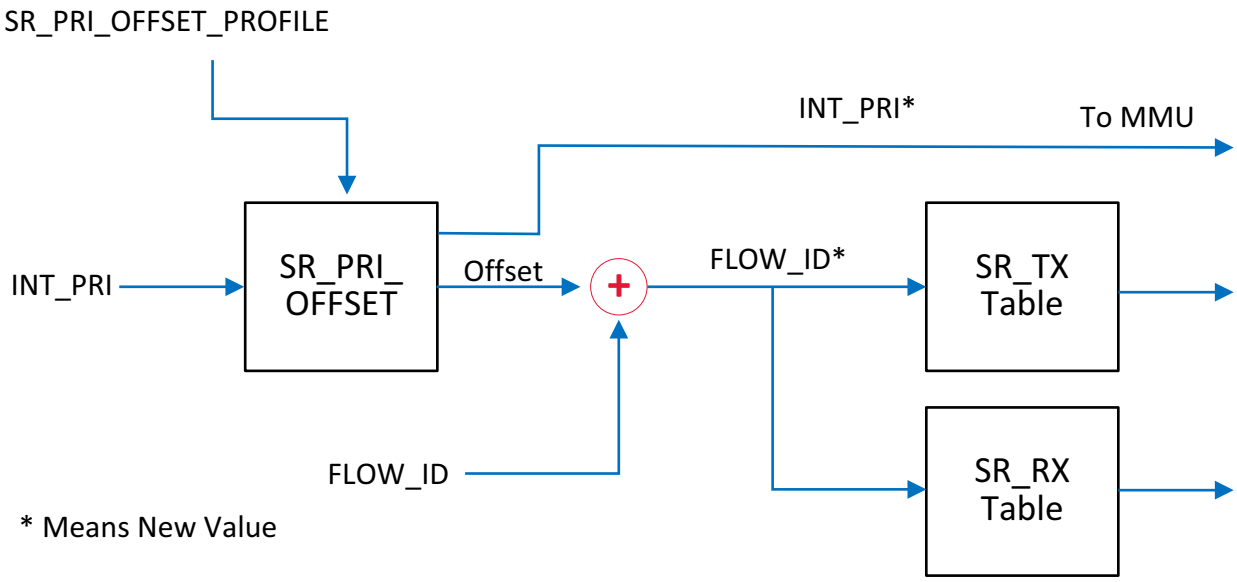
The sequence number TX is reset to 0 by software when required, such as when a new flow is detected.

6.3.7.4 SR FLOW_ID and INT_PRI Assignment

As the following figure shows, the initial INT_PRI assigned by the pipeline is used along with SR_PRI_OFFSET_PROFILE to lookup the SR_PRI_OFFSET table and derive the offset to be added to FLOW_ID. The final FLOW_ID is then used to index the SR_TX and SR-RX tables.

The SR_PRI_OFFSET table can also assign a new INT_PRI, if required. This new INT_PRI can then be used by the MMU for queuing.

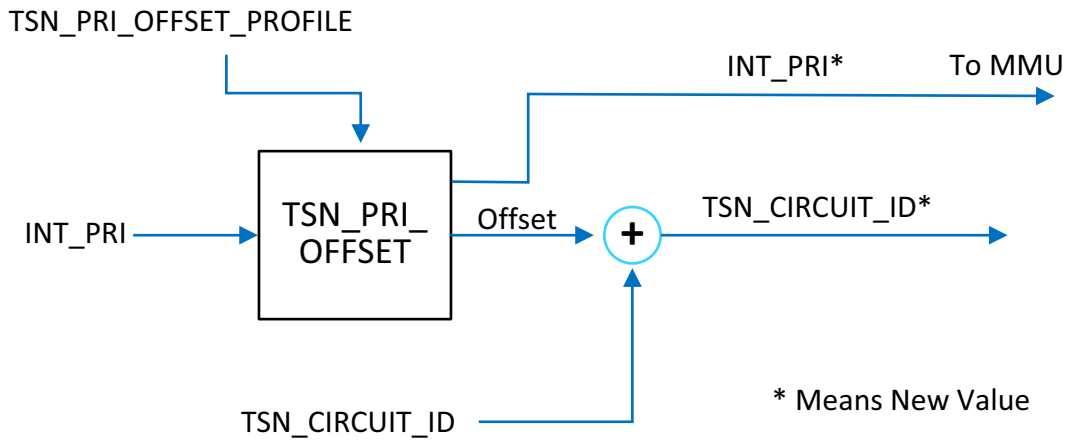
Figure 77: FLOW_ID Priority Offset



6.3.7.5 TSN_CIRCUIT_ID and INT_PRI Assignment

The TSN_CIRCUIT_ID offset can be derived base on the NT_PRI and TSN_PRI_OFFSET_PROFILE indexes as shown in the following figure. In addition, a new INT_PRI can be assigned per TSN_CIRCUIT by using the TSN_PRI_OFFSET table.

Figure 78: TSN_CIRCUIT_ID Priority Offset



NOTE: When a packet is both an SR packet and a TSN packet, the INT_PRI must be configured the same as it is in the TSN_PRI_OFFSET table and SR_PRI_OFFSET table for that packet.

6.3.7.5.1 Duplicate Filtering

The BCM56070 supports duplicate filtering in the ingress pipeline by using the SR_RX table and in the egress pipeline by using the EFP. This means the duplicate filtering in the ingress pipeline is per SR-flow filtering. However, in some cases, the duplicate filtering may be an egress port property.

Example: There can be cases that an HSR ring forwards packets to two Interlinks, where one interlink is in duplicate discard mode, and the other interlink is in duplicate accept mode. The BCM56070 allows for both behaviors.

For ingress filtering, the following three fields in the SR_RX table control the forwarding of duplicate packets and packets that fall outside of the SN_WINDOW or are out of order:

- ACCEPT_DUPLICATE_PACKET
- ACCEPT_IN_DROP_WINDOW
- DROP_OUT_OF_ORDER

However, the IFP can override the duplicate ACCEPT_DUPLICATE_PACKET by using the following control:

- DROP_SR_FRAME (2 bits)
  - 0 = Noop
  - 1 = Drop if the packet is an SR duplicate
  - 2 = Do not drop if the packet is an SR duplicate
  - 3 = Reserved

For egress filtering, when the packet is a duplicate, the EFP has the following control for forwarding or dropping the duplicate packets:

- DROP_SR_FRAME
  - 0 = Noop
  - 1 = Drop if the packet is an SR duplicate
  - 2 = Do not drop if the packet is an SR duplicate
  - 3 = Reserved

6.3.7.6 PTP Forwarding

6.3.7.6.1 Domain ID Filtering

IEEE 1588v2 supports multiple timing domains. The timing domain is encoded in the IEEE 1588v2 message header in the Domain Number field (8-bit), as shown in the following table.

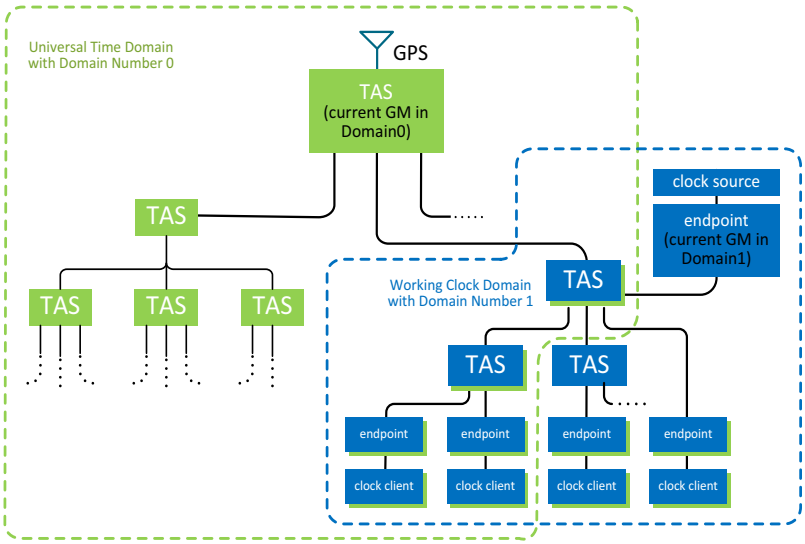
The switch requires supporting four timing domains. For industrial networks, usually one domain is the universal clock, and other domains are working clocks that are not necessarily synchronized to the universal clock.

Table 8: IEEE 1588v2 Message Header

Bits								Octets	Octets
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**		
transportSpecific				messageType				1	0
reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
reserved								1	5
flagField								2	6
correctionField								8	8
reserved								4	16
sourcePortIdentify								10	20
sequenceID								2	30
controlField								1	32
logMessageInterval								1	33

The following figure shows an industrial network with two timing domains.

Figure 79: Multiple Timing Domains Example



As the figure shows, any port can be in one or more timing domains. A strict requirement is that PTP packets from one domain should not leak to the other domain. This means a PTP packet should only be forwarded to the ports that support the domain ID encoded in the PTP message.

To achieve timing domain ID filtering, the specification allows each port to be configured with up to four domain IDs. The packet's domain ID is compared to the domain ID of the ingress and egress ports, and if the ingress or egress ports do not support the packet's domain ID, the condition is reported to the IFP and EFP, and the IFP and EFP can drop packets that are received from or transmitted to the wrong PTP domains.

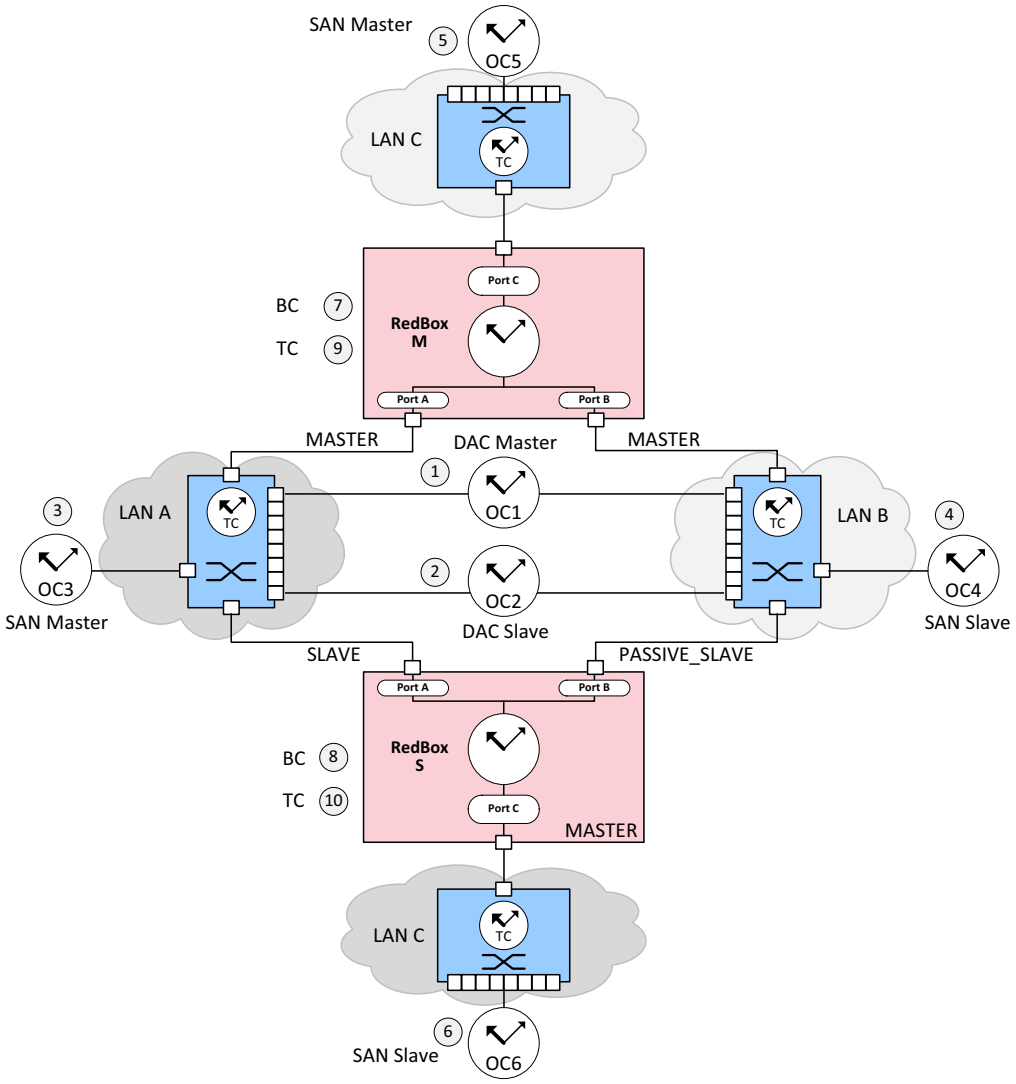
In addition, the packet and port domain ID are fed to the IFP and EFP as keys and, alternatively, IFP and EFP rules can be created to drop packets that are entering wrong domain.

6.3.7.6.2 IEEE 1588v2 and IEEE 802.1AS Forwarding

PRP, HSR, and IEEE 802.1CB networks should support IEEE 1588v2 and IEEE 802.1AS timing synchronization. This includes TC clocking and means there may be a seamless redundancy network between the master and slave clock as shown in Figure 80.

In this example, a PRP network is between the master and slave clocks. and the master and slave clocks are SAN. Several different methods are available to forward the PTP messages over the seamless redundancy network and are described in this section. However, an important point is that for IEEE 1588v2 and IEEE 802.1AS timing synchronization, the SR duplicate discard should not be used. The reason is that in seamless redundancy, the two copies of a packet that traverse different paths have different delays. Therefore, the slave cannot select the PTP messages based on the first PTP message that it receives (as in seamless redundancy). Instead, the slave should only use PTP messages from one direction (LAN_ID) and discard the PTP messages from the other direction (LAN_ID). If a failure occurs, the slave can then switch receiving from one LAN_ID to another LAN_ID.

Figure 80: SAN Master and Slave Outside of PRP Network



The following sections describe the two main methods to forward PTP messages over a seamless redundancy network.

6.3.7.6.2.1 Method One

In this method, the master and slave are capable of interpreting SR tags. The ingress Red-box that receives a PTP packet duplicates the PTP (IEEE 1588v2) packet as for any data packet. If the PTP packet is injected by CPU, this specification assumes that the pipeline performs packet duplication and creates two copies of the PTP packet.

NOTE: The BCM56070 assumes the PTP flow maps to its own dedicated FLOW-ID.

When a remote Red-box receives PTP messages that need termination or copy-to-CPU, it forwards both duplicate copies to the CPU and keeps the SR tag. The slave processes the PTP messages coming from a single direction (as indicated by the LAN_ID in the packet SR tag) and discards packets received from the other LAN_ID.

6.3.7.6.3 Method Two

In this method, the master and slave are not capable of interpreting SR tags. Therefore, another method is used to convey to the LAN-ID that a packet is received from the master or slave. Appendix A of the *HSR/PRP Standards* indicates the Source Port Identity (SPI) field of PTP packets conveys the LAN-ID information to the master or slave (or both). The Red-box is responsible for mapping the LAN-ID to the SPI, and vice versa.

Two possible cases are supported:

- Only the Red-box connected to the master clock manipulates the source port identity field of the PTP message.
- Only the Red-box connected to the slave manipulates the source port identity of the PTP message.

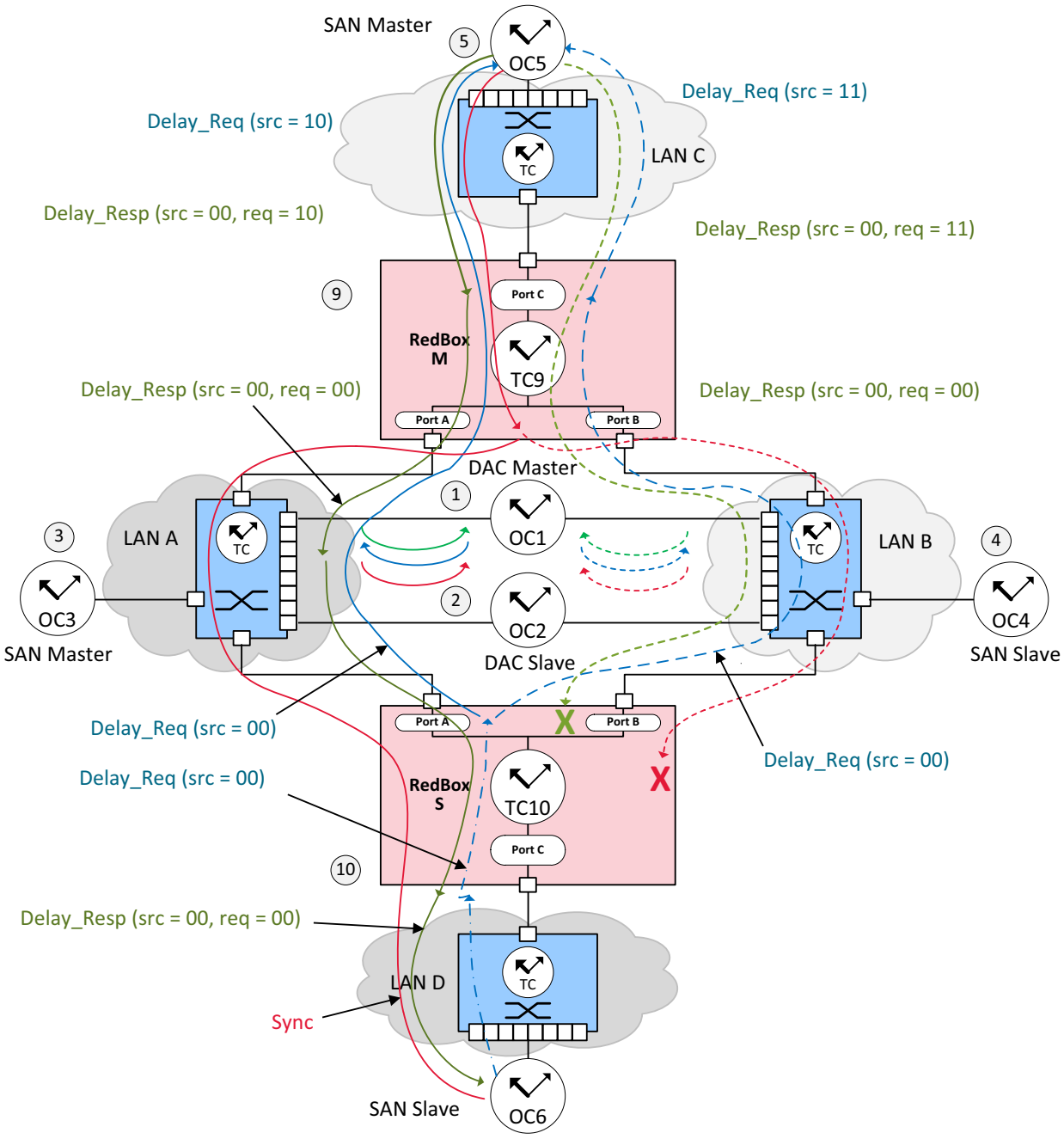
This section uses the following terminology:

- src = xy is a 2-bit field where x represents bit 13, and y represents bit 12 of the SPI.
- req = zw is a 2-bit field where z represents bit 15, and w represents bit 14 of the SPI.
- 10 represents LAN-A, and 11 represents LAN-B.

6.3.7.6.3.1 Case One

Case one is shown in Figure 81. The master sends sync message to the slave. Red-box M simply duplicates the packet to LAN-A and LAN-B without modifying the SPI. Red-box S does not perform a duplicate discard; instead, it always discards packets received from LAN-B. This means that it discards the sync message received from LAN-B and forwards the sync message received from LAN-A to the slave.

Figure 81: Master Clock Red-box Can Only Manipulate Source Port Identity



The slave responds with Delay_Req (src = 00). Red-box S duplicates that message to LAN-A and LAN-B without modifying the SPI. Red-box M, upon receiving the Delay_Req (src=00) from LAN-A, modifies it to Delay_Req (src = 10). Upon receiving the Delay_Req (src = 00) from LAN-B, Red-box M modifies it to Delay_Req (src = 11).

The master received both Delay_Req (src = 01) and Delay_Req (src = 11) from the Interlink of Red-box M. The master then responds to both messages separately, but it manipulates the SPI of the Delay_Resp by copying the SRC field to the REQ field, and then the master sets SRC = 00. Therefore, the master responds to Delay_Req (src = 01) by generating Delay_Resp (src = 00, req = 01) and responds to Delay_Req (src = 11) by generating Delay_Resp (src = 00, req = 11). Red-box M, upon receiving the Delay_Resp, checks the SRC of the SPI, and if src = 10, it sends the packet only to LAN-A. If src = 11, Red-box M sends the packet only to LAN-B (no SR duplication). Red-box M also changes the src = 00 and req = 00. Red-box S, as usual, discards the Delay-Resp received from LAN-B and forwards the Delay_Resp received from LAN-A to the slave.

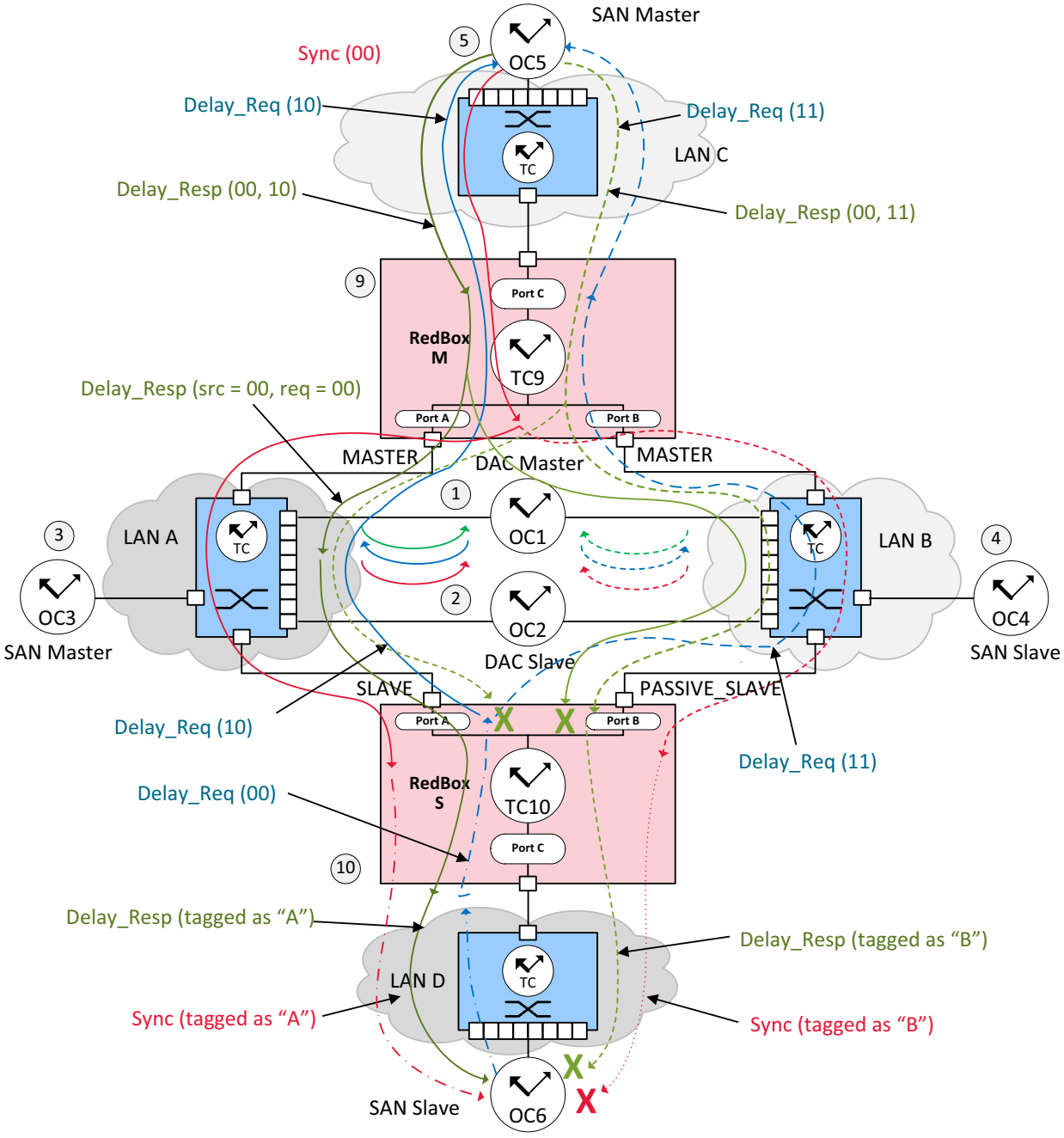
The following special behaviors are required by the Red-box:

- Not performing SR duplicate discard
- Discarding packets received from LAN-B unconditionally
- Changing SPI.src when forwarding to the Interlink based on the LAN_ID of the received packet
- Forwarding the packet received from the Interlink to LAN-A or LAN-B, based on the SPI.req
- Changing the SPI.req of the packet to req = 00 when forwarding the packet received from the Interlink to LAN-A or LAN-B

6.3.7.6.3.2 Case Two

Case two is shown in Figure 82. the master sends sync message to the slave. Red-box M simply duplicates the packet to LAN-A and LAN-B without modifying the SPI. Red-box S does not perform duplicate discard and forwards both sync messages received from LAN-A and LAN-B to the slave. Red-box S also sets the SPI.src = 10 for sync messages received from LAN-A, and it sets the SPI.src = 11 for sync messages received from LAN-B.

Figure 82: Slave Clock Redbox Can Only Manipulate Source Port Identity



The slave responds with Delay_Req (src = 00). Red-box S duplicates that message to LAN-A and LAN-B and models the SPI. Red-box S sets Delay_Req (src = 10) when it sends the delay request to LAN-A and it sets Delay_Req (src = 11) when it sends delay request to LAN-B. Red-box M does not do supplicate discard and sends both Delay_Req (src = 10) and Delay_Req (src = 11) to the master.

The master responds to each Delay_Req (src = 10) by generating Delay_Resp (src = 00, req = 10) and responds to Delay_Req (src = 11) by generating Delay_Resp (src = 00, req = 11). Delay_Resp (src = 00, req = 10) and Delay_Resp (src = 00, req = 11) get duplicated by Red-box M as usual, which results in four Delay_Resp (src = 00, req = 10) messages in the SR network.

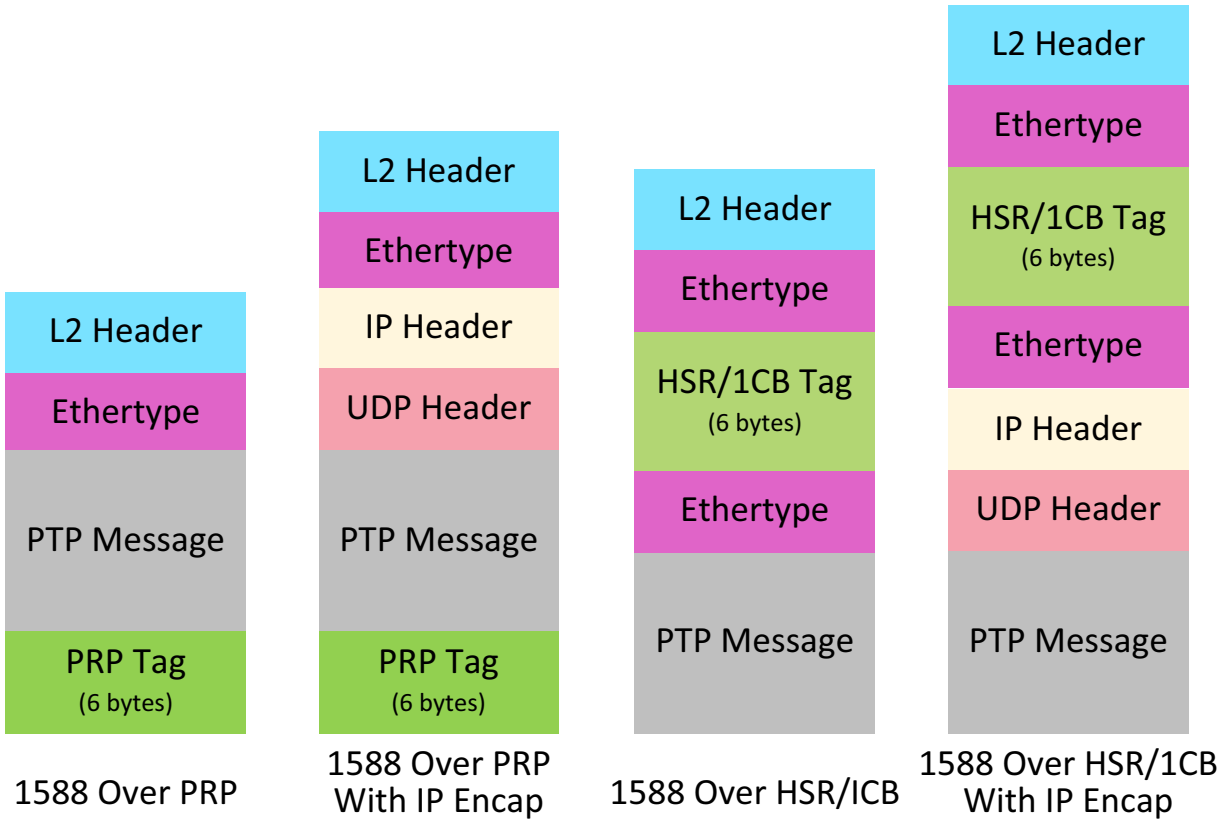
Red-box S discards Delay_Resp (src = 00, req = 10) from LAN-B and discards Delay_Resp (src = 00, req = 11) from LAN-A. Red-box S also changes Delay_Resp (src = 00, req = 10) to Delay_Resp (src = 10, req = 00), and it changes Delay_Resp (src = 00, req = 11) to Delay_Resp (src = 11, req = 00).

To support the SPI manipulation by the Red-box, this specification feeds the SPI field to the IFP and EFP as a key. In addition, new IFP and EFP actions are defined to change any bits in the SPI by using a mask. The assumption is that the egress parser can parse the packet for the required fields.

6.3.7.7 PTP and IEEE 802.1AS Transparent Clocking (TC)

The BCM56070 supports 1-step and 2-step TC for the encapsulations as shown in the following figure.

Figure 83: PTP Over HSR, PRP, and IEEE 802.1CB



6.3.7.7.1 TC Support in HSR and IEEE 802.1CB

To support TC and OC in HSR, the following requirements must be met:

- The parser is required to identify the PTP messages encapsulated in HSR and IEEE 802.1CB.
- Egress Packet Modification (EPMOD) is required to update the Correction Field for PTP packets with HSR and IEEE 802.1CB encapsulation.

6.3.7.7.2 TC Support in PRP

TC is not supported in pure PRP nodes because PRP switching is not allowed between PRP ports. However, in PRP-HSR interworking, TC may be supported between PRP and HSR ports.

6.3.7.8 SAN Forwarding in PRP

It is possible to receive non-RCT tagged packets on a PRP port coming from a SAN in LAN-A or LAN-B. Also, when forwarding to a SAN in a PRP network, the Red-box should not duplicate or add an RCT tag.

This RX function can be achieved by assuming that a PRP port is allowed to receive non-PRP packets.

The TX function is achieved by introducing a field in the L2 table called SAN. When a DA lookup indicates SAN = 1, the packet should be forwarded normally without duplication and without adding an RCT tag. The BCM56070 uses the IFP and EFP to enforce this behavior.

6.3.7.9 Red-Box TX Direction

This section describes the TX direction flow and associated lookups in the BCM56070. The TX direction is the direction when packets are received from the Interlink or Link Layer and are duplicated and forwarded to HSR, PRP, and 1CB ports. Forwarding for unicast and multicast is the same.

When an Ethernet frame is received, the PORT table, VLAN table, and ING_EGR_PORT table provide the seamless redundancy parameters to the IFP. In addition, MAC-DA and MAC-SA lookup is done in the L2_ENTRY table, and the lookup hit status is fed to the IFP.

Based on IFP configurations, the IFP provides the action to drop or forward the packet, and possibly perform a copy-to-CPU action. If the decision is to forward, the IFP enables the SR_TX table to increment the sequence number and pass the resulting SR_NEXT_SN to the EP, as well as PKT.LAN_ID, PKT.NET_ID, and PKT.SR_TYPE.

The L2_ENTRY (MAC-DA) lookup provides L2MC, where the L2MC is composed of the two HSR, PRP, or 1CB ports to which to replicate the packet. If the L2_ENTRY (MAC_SA) lookup fails, the IFP forwards the packet to the CPU for MAC learning.

The FLOW_ID selection source is based on the PORT table configuration. The FLOW_ID is either derived from the L2_ENTRY (MAC-SA) lookup or the L2_ENTRY (MAC-DA) lookup, or it is derived from VFP rules.

In the EP, the EGR_PORT, EGR_ING_PORT, and EGR_VLAN tables provide the SR parameters to the EFP. In addition, the SR_NEXT_SN is provided to EPMOD. The EFP then decides to forward the packet and add an HSR, PRP, or 1CB tag to the packet.

6.3.8 Red-Box RX Direction

This section describes the RX direction flow and associated lookups in the BCM56070. The RX direction refers to the direction from the HSR, PRP, or 1CB ports to the Interlink or Link Layer, and discarding duplicate packets.

When SR-tagged packets are received from SR ports, the flow is very similar to the TX flow. However, there are a number of differences.

When an HSR, PRP, or 1CB frame is received, the following parameters are fed to the IFP:

- Seamless Redundancy parameters (see Section 6.3, Seamless Redundancy) from the PORT table, VLAN table, and ING_EGR_PORT table
- MAC-DA and MAC-SA lookup in the L2_ENTRY table hit status
- PKT.LAN_ID and PKT.NET_ID
- SN_HISTORY lookup results

6.3.8.1 SR_PORT_ROLE and SR_TYPE

The BCM56070 introduces the following two properties to each port:

- SR_PORT_ROLE
  - Interlink
  - SR Port
- SR_TYPE
  - Ethernet
  - PRP
  - HSR
  - IEEE 802.1CB

SR_PORT_ROLE tells the chip whether that port is an Interlink or SR duplicate port. SR-Type specifies which types of packets to expect on that port (Ethernet, HSR, PRP, or IEEE 802.1CB).

In a Red-box, the port that has PORT_ROLE_Interlink is usually the port that has no knowledge of SR and is a pure Ethernet port (SR_Type = Ethernet). However, for an HSR or PRP interworking box, the port connected to the PRP network also has PORT_ROLE = Interlink but SR_TYPE = PRP.

In general, when a PORT_ROLE = Interlink and PORT_TYPE != Ethernet, the Red-box is an interworking box and should translate the SR tags between its ports.

6.3.8.2 SR_TX and SR_RX

The SR_TX table generates sequence numbers for TX packets received from the Interlink or Link-Layer. It is logically located after the IFP.

The SR-RX table checks the duplicate discard function and sequence number tracking. It is logically located in parallel to the IFP, and the IFP cannot prevent the SR_RX and its related SN_WINDOW from being updated by a new packet.

It is critical to determine whether a packet should follow the SR_TX flow or SR_RX flow. In general, this is a port property (not a packet property). If a port is SR-enabled, and it is an Interlink that expects normal Ethernet packets, then the packet should follow the SR_TX flow. However, when the port is an SR port, and the port is an HSR, PRP, or 1CB port, then the packet should follow the SR_RX flow. Otherwise, the packet should not follow either the SR_R flow or the ST-TX flow.

6.3.8.3 S-VLAN and Default L2.BITMAP

The BCM56070 has 64 ports and can support many Red-boxes. These Red-boxes must be isolated from each other so the traffic does not leak from one Red-box to another Red-box.

The BCM56070 uses the S-VLAN to isolate the Red-boxes. Each Red-box has a different S-VLAN. Packets entering the chip from any port are assigned an S-VLAN that depends on which Red-box they belong to.

One of the advantages of assigning an S-VLAN to a Red-box is for unicast packets with an unknown MAC-DA. In the HSR or PRP, this assignment requires flooding unknown unicast packets. Having a separate S-VLAN allows a separate flood domain by configuring a separate default L2.BITMAP for those S-VLANs.

NOTE: When an Interlink is shared between two or more Red-boxes, the Red-boxes are considered to be part of the same broadcast domain and should be assigned the same S-VLAN.

IEEE 802.1CB is a fully provisioned and configurable network, and flooding of unknown unicast packets should not be done. Instead, unknown unicast packets should be dropped. This is accomplished by configuring the default L2.BITMAP for all IEEE 802.1CB Red-boxes as zero, with no member port.

For 1CB, different S-VLANs can be used for LAN-A and LAN-B duplicate ports. For HSR or PRP, two duplicate ports can be part of the same S-VLAN only.

6.3.8.4 Packets Transmitted from or Received by the CPU

The CPU can inject packets into an SR network using front-panel forwarding mode (not SOBMH or HiGig modes). When this occurs, the pipeline should be able to forward packet normally and assumes the packets came from Port = 0, where Port = 0 is configured as an Interlink port.

The pipeline should be able to duplicate these packets, increment the SR-TX sequence number, and add an SR tag to the packet.

Similarly, when a packet is sent to the CPU (Port = 0), it should be forwarded as though it is being forwarded to an Interlink, provided that Port = 0 is configured as an Interlink.

NOTE: COPY_TO_CPU is different from forwarding to the CPU. In general, copy-to-CPU keeps the original packet encapsulation, including the SR tag.

6.3.8.5 LAN_ID, NET_ID, and PATH_ID

This section describes the relationship between LAN_ID, NET_ID, and PATH_ID, as well as their encoding. The HPRP RCT tag uses a 4-bit field called LAN_ID to encode the LAN_ID with the following:

- LAN_ID = 1010 = LAN-A
- LAN_ID = 1011 = LAN-B

The HSR PATH_ID uses a 4-bit field called Path_ID to encode both NET_ID and LAN_ID.

The following table shows the HSR Path_ID encoding. As the table shows, that packet generated within an HSR network has an encoding of 0000 or 0001. Other encodings are for connecting the seven PRP networks to an HSR.

NOTE: When a packet is forwarded from an HSR network to the PRP, the LAN_ID in the RCT tag is always changed to one of the following:

- LAN_ID =1010 (LAN-A)
- LAN_ID =1011 (LAN-B)

Table 9: HSR Path_ID Encoding

HSR Path_ID	NET_ID	LAN_ID	Comments
0000	0	A	From HSR node
0001	0	B	From HSR node
0010	1	A	From PRP Network 1
0011	1	B	From PRP Network 1
0100	2	A	From PRP Network 2
0101	2	B	From PRP Network 2
0110	3	A	From PRP Network 3
0111	3	B	From PRP Network 3
1000	4	A	From PRP Network 4
1001	4	B	From PRP Network 4
1010[a]	5	A	From PRP Network 5
1011[a]	5	B	From PRP Network 5
1100	6	A	From PRP Network 6
1101	6	B	From PRP Network 6
1110	7	A	From PRP Network 7
1111	7	B	From PRP Network 7

a. Although these values look the same as the LAN_ID in the PRP network, they are unrelated and have independent codings from the PRP network.

The NET_ID and LAN_ID terms are defined as follows:

- NET_ID indicates the three MSB bits of the 4-bit field in the HSR PATH_ID and in the RCT LAN_ID.
- LAN_ID indicates the LSB bit of the 4-bit field in the HSR PATH_ID and in the RCT LAN_ID

This means when an SR packet enters the ingress pipeline, the PKT.NET_ID is extracted from the upper 3 bits of the HSR PATH_ID or the upper 3 bits of the PRP LAN_ID. The PKT.LAN_ID is extracted from the LSB bit of the HSR PATH_ID or the LSB bit of the PRP LAN_ID.

When an SR packet exits the egress pipeline, the HSR or PRP tag is constructed as follows:

```
HSR.PATH_ID = (NET_ID, LAN_ID)
PRP.LAN_ID = (NET_ID, LAN_ID) //NET_ID must be configured as “101”
```

6.3.8.6 Mirroring

Ingress and egress mirroring is supported in the BCM56070. True egress mirroring is not supported.

For HSR and IEEE 802.1CB, the mirroring is as usual. For PRP, the ingress mirroring is as usual, but the egress mirroring must be done in the second pass of the pipeline, and it is necessary to use a loopback port when mirroring is enabled and to provide a bitmap for the egress port instead of for the original ingress port.

6.3.8.7 Unexpected SR Frame

The BCM56070 detects unexpected SR frames that are received from a port. Unexpected frames are defined as follows:

- When an HSR or IEEE 802.1CB frame is received on a PRP port
- When an HSR frame is received on an IEEE 802.1CB port
- When an IEEE 802.1CB frame is received on an HSR port

NOTE: When the PRP packet is received on an HSR or IEEE 802.1CB port, it will not be an unexpected SR frame because the PRP trailer cannot be identified with complete confidence because some data may look like a PRP trailer.

6.3.8.8 MTU and STU Error

These specifications can check the maximum (MTU) and minimum (STU) frame size and can also check when a frame does not conform to the proper format. If this occurs, the BCM56070 generates an MTU or STU error and increments the corresponding error counters.

However, it is not possible to copy packets to the CPU with an MTU or STU error because the MTU or STU check is done at the EOP, and by that time, the packet is already queued in the egress queue in the MMU, so it is too late to send a copy-to-CPU queue.

6.3.8.9 Wrong LAN Error

The WRONG_LAN_ERROR is set when the LAN_ID in the packet does not match the LANID of the actual packet. WRONG_LAN_ERROR packets should *not* be dropped and should be processed as usual, including the duplicate discard processing.

WRONG_LAN_ERROR packets increment the WRONG_LAN_ERROR counter and set the proper WRONG_LAN bit in the SR_TX table.

6.3.8.10 DLF Forwarding

In HSR and PRP nodes, the unknown unicast or multicast packets must be flooded to the VLAN domain. However, in IEEE 802.1CB unknown unicast or multicast must be dropped.

Flooding is independent of SR-TX or SR-RX processing. This means that the flooded packet will go under SR processing and may generate SR duplicate packets.

Figure 84: First Example of DLF Flooding

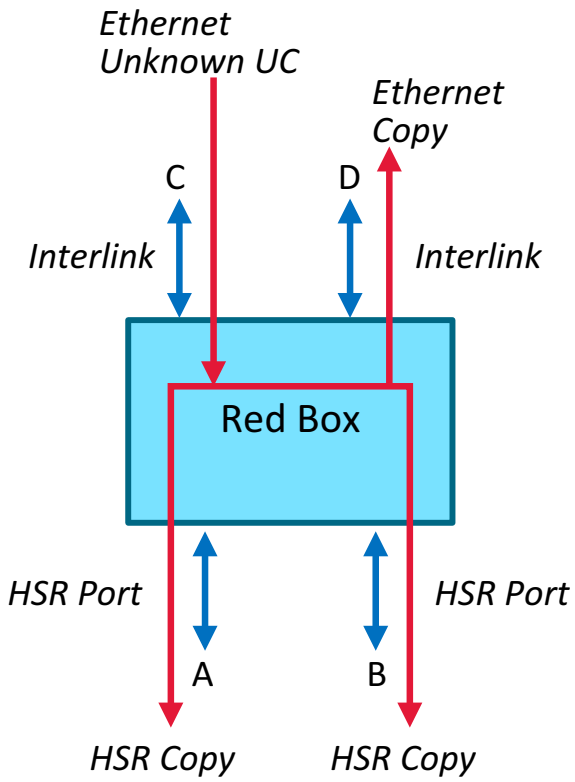
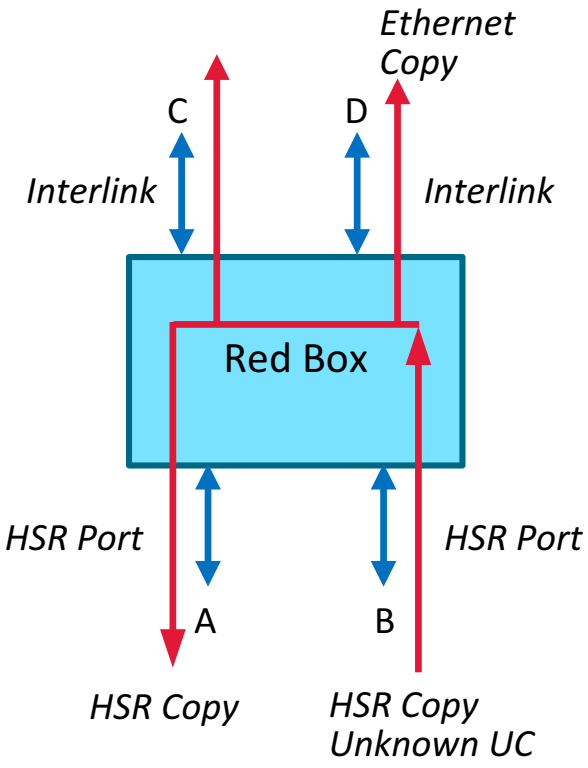


Figure 84 shows an example of a Red-box with two Interlink ports (C and D) and two HSR ports (A and B). When an Ethernet packet with an unknown DA is received from port C, it is flooded to all ports, including ports A, B, and D. The packet goes through the ST-TX flow and gets a sequence number assigned to it. Copies of the packet that are forwarded to ports A and B have an HSR tag. The copy that is forwarded to port D does not have an HSR tag. The EFP decides whether or not to add a tag to the packet based on the egress port property.

Figure 85: Second Example of DLF



In Figure 85, the unknown unicast packet is received from HSR port B. This packet goes through the SR-RX flow and duplicate discard process, and it is flooded to ports A, C, and D. The copy sent to port A has an HSR tag but the copies sent to ports C and D have no HSR tag.

An infinite loop will not happen because the node that injected the packet into HSR ring will filter the packet using SA-Proxy filtering.

6.3.8.11 Ethernet OAM

SR networks may run Ethernet OAM. However, Ethernet OAM packets should not have an SR tag.

For PRP, the Ethernet OAM packets are copied to the OLP or CPU in the first pass, so they are not counted in the PORT_SR_RX_COUNTER or FLOW_SR_RX_COUNTER.

6.3.8.12 SR Counters

The BCM56070 includes PORT and FLOW counters for RX and TX directions. It includes both error counters and non-error counters. Each counter can have a configurable threshold, and when the counter threshold is exceeded and, if configured, the BCM56070 can optionally interrupt the CPU and send a copy of the packet to the CPU.

The BCM56070 defines a set of eight counters per-port and per-flow. This means it can count only eight events in the RX direction (ingress pipeline) and four events in the TX direction (Egress pipeline). However, there may be more than eight possible events that can occur in the RX direction, and there are more than four possible events that can happen in the TX direction. This BCM56070 has a configurable way of assigning events to counters so that users can choose which events are counted.

All counters are incremented at EOP. The SR counters have the following restrictions:

- Errors that happen at EOP cannot cause the packet to be coped to the CPU. These errors include the following:
  - MTU_ERROR
  - STU_ERROR
  - TAG_ERROR

  A copy-to-CPU is not permitted because the BCM56070 does not support EP-redirect.
- When a counter is incremented, and the counter threshold is reached, the next packet that increments the counter will interrupt the CPU and can be copied to the CPU. However, the packet that caused the counter to reach the threshold cannot be copied to the CPU because the interrupt and copy-to-CPU happen at SOP, but counters are incremented at EOP.

  Exceptions to this rule are MTU_ERROR and STU_ERROR counters, which send an interrupt to the CPU at EOP, so the packet that caused the counter to reach the threshold will interrupt the CPU.

6.4 Time-Aware Filtering and Policing

This section discusses the time-aware filtering and policing features in the BCM56070

6.4.1 Overview

Time Sensitive Networking (TSN) deals with networking of elements or end points which are sensitive to or aware of time. A set of IEEE standards have been defined to support time sensitive networking.

Time Aware Filtering and Policing (TAF) is a feature of TSN. TAF is defined in IEEE standard 802.1Qci/D1.4. TAF ensures that frames belonging to a stream are allowed to pass through a network element only during a specified time slot.

The BCM56070 supports stream and flow identification, as well as per stream filtering. It also has support for 4K dual bucket Service Meters and Policers, in compliance with MEF 10.3. The BCM56070 also supports time-aware frame gating to provide full support for Time Aware Per-Stream Filtering and Policing in compliance with IEEE 802.1Qci.

IEEE 802.1Qci deals with frame filtering, time aware frame gating, metering and policing, and service class selections for frames of specific data streams.

NOTE: TAF supports a single-chip system configuration and does not support systems in multi-chip using HiGig stack and HiGig proxy.

The BCM56070 supports the following:

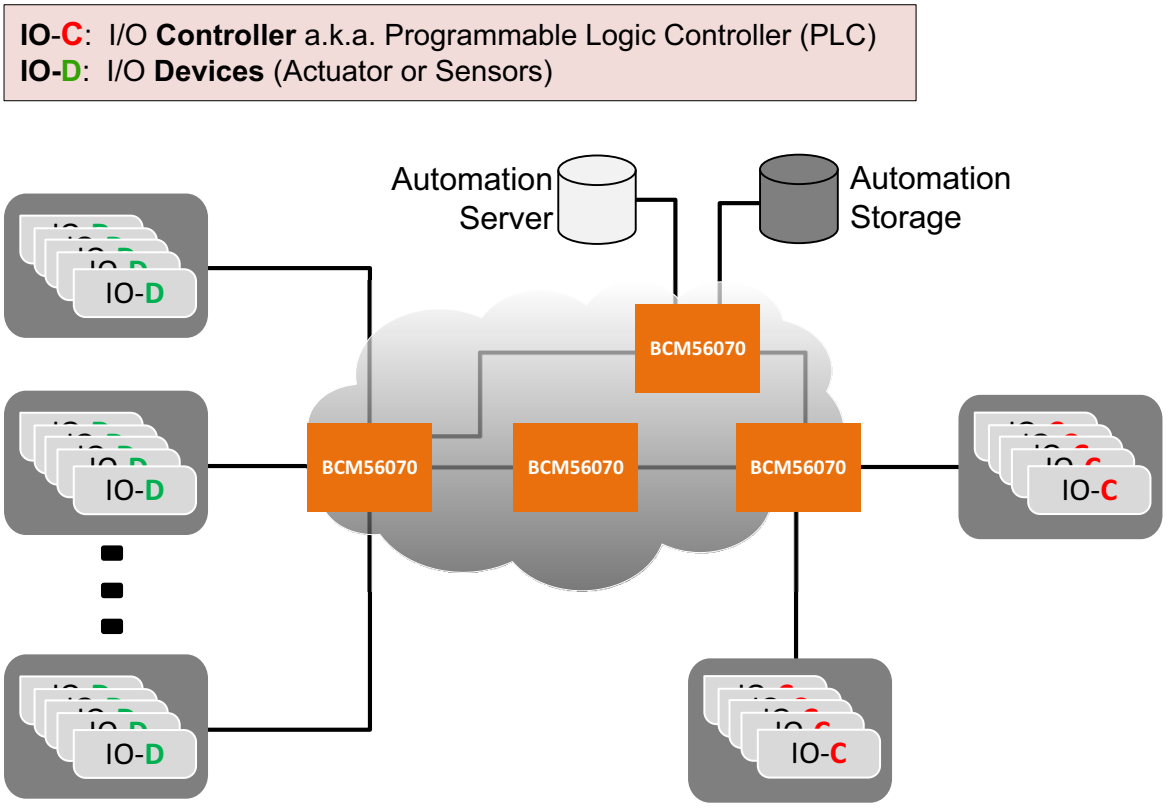
- 128 stream gates for TAF function.
- Streams can be assigned by the VFP, VXLT, L2 lookup, and ingress port.
- 16 entries in the gate control table per TAF gate.
- Any number of ports or streams can be mapped to a TAF gate.
- Maximum byte limit that can pass through the gate during open state.
- Gate assignment based on the preemptable or express traffic type from the MAC and PHB.

6.4.1.1 Network Level Flow

While Ethernet has been adopted for time-noncritical industrial applications, there are hurdles in using the technology for time-critical applications due to the inherent reliability concerns and non-deterministic latencies through the network. This has changed over the last few years with the enhancements in Ethernet standards (for example, IEEE1588 and AVB or TSN), which enable further penetration of Ethernet technology in almost every application of industrial automation.

The following figure shows an industrial automation environment with connectivity across sensors and actuators, controllers, and an automation server. Industrial automation environments consists of distributed control systems containing a large number of interconnected devices that exchange data periodically through communication networks.

Figure 86: Industrial Automation Environment



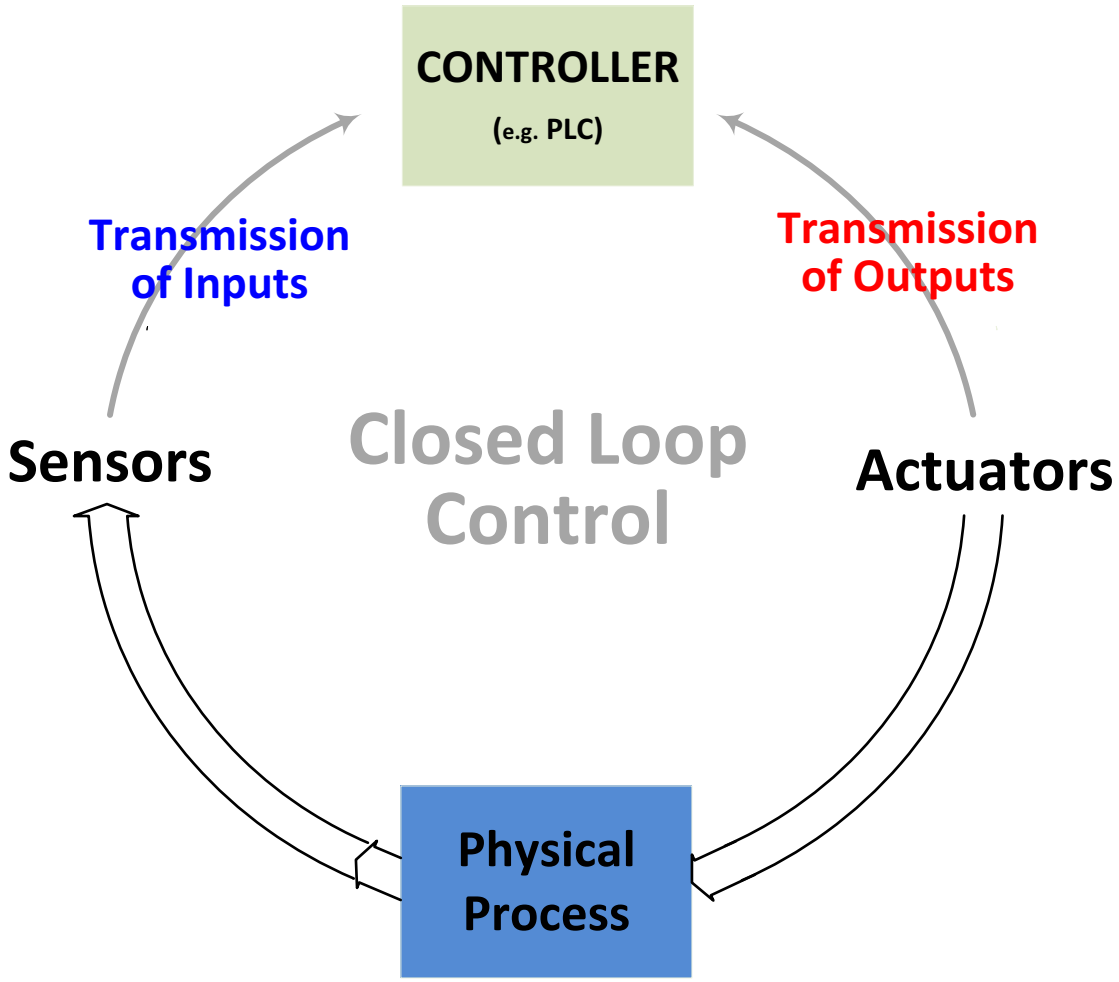
The system implements closed control loops between the controller and the devices, such as actuators and sensors.

The control loop or control cycle is divided into two parts.

- Communication cycle
  - Time during which controller and the devices exchange data
  - Requires deterministic packet scheduling
- Compute cycle
  - Time during which controller processes the data
  - No time sensitive data is exchanged and is ideal for the best effort traffic

In a given industrial automation environment, there can be multiple control loops running concurrently. Using Time Aware Gating, time slotted communication cycles can be setup across these concurrently running control loops. This not only helps improve the bandwidth utilization of these industrial communication networks, but also reduces the impact of malfunctioning devices by isolating them quickly.

Figure 87: Control Loop between Controller, Actuators, and Sensors



6.4.1.2 System Level Flow

In a switching system, Time Aware Filtering and Policing needs to be enabled only in the ingress BCM56070, which is directly connected to the sensors and actuators. Subsequent BCM56070 devices in a system perform normal packet forwarding.

6.4.1.3 Chip Level Packet Flow

This section describes the chip level flow for Time Aware Stream Gating function. A TAF stream is identified by SR_FLOW_ID implemented to support seamless redundancy (IEEE 802.1CB) in BCM56070.

6.4.2 TAF and Seamless Redundancy Interactions

This section describes the interaction handling between 802.1Qci and 802.1CB functions if both are enabled on same BCM56070 chip.

- If SR_RX flows are enabled with TAF on a given BCM56070, enabling MaxBytes check on a given TAF gate will not work accurately if all the flows mapped to a Qci gate do not have SR_RX enabled. This is because SR_RX-enabled flows will receive duplicate packets and there will be a double byte counting for the corresponding TAF gate. Non SR_RX-enabled flows mapped to same TAF gate will not have duplicate packets. Because of this, do not enable a MaxBytes check on a TAF gate if all the flows mapped to a TAF gate do not have SR_RX enabled.
- If all the flows mapped to a TAF gate have SR_RX flow enabled, then accurate byte counting can be achieved for that gate by doubling the MaxBytes count that can pass through the gate when it is open.
- If SR_TX is enabled with TAF, the SR_TX sequence number is incremented at the SOP. However, the TAF gate could decide to purge the packet at EOP. If that occurs, there will be sequence number of holes in the SR_TX flow that must be handled at the remote receiver. TAF gate MaxBytes check can be enabled with SR_TX flows.
- Each TAF gate counts packet pass and drop or purge decisions in the dedicated PASSED_PACKET and DROP_PACKET counters. These counters are independent of the packet drop decisions in SR or IFP lookups, or any other lookup later in the pipe. SR-related drops are counted in dedicated SR drop counters. Similarly, IFP lookup drops are counted in IFP drop counters.

6.4.3 TAF Frame Statistics Counters

The IEEE P802.1Qci specification describes the following frame statistics counters:

- A count of frames matching both the stream_handle and priority specifications:
  - 802.1Qci standard requires using stream_handle provided by the stream identification function implemented in the 802.1CB function. In the BCM56070, SR flex counters are implemented to count frames matching both the stream_handle and priority specifications using final_sr_flow_id.
- A count of frames that passed the stream gate:
  - For each TAF gate, there is a dedicated counter called TAF_GATE_PASSED_PACKET_COUNTER for counting frames that passed the gate across all the streams assigned to that gate.
  - A new bit, TAF_STREAM_GATE_PASS, was added in the SR_ING_COUNTER_PROFILE to count the frames using SR flex counters that passed the gate for that stream.
- A count of frames that did not pass the stream gate:
  - For each TAF gate, there is a dedicated counter called TAF_GATE_DROP_PACKET_COUNTER to count frames that were marked TAF_DROP or TAF_PURGE through the gate.
  - A new bit called TAF_STREAM_GATE_CLOSE was added in the SR_ING_COUNTER_PROFILE to count the per-stream frames using SR flex counters that were either dropped or purged because the gate was in the *close* state when the frame passed through the gate.
  - A new bit called TAF_STREAM_GATE_NOBYTE was added in the SR_ING_COUNTER_PROFILE to count the per-stream frames using SR flex counters that were either dropped or purged because the gate was in the *open* state, but the gate's BYTE_LEFT value was less than the value of PACKET_BYTE_COUNT.
- A count of frames that exceeded the maximum SDU size:
  - A new bit called TAF_MTU_PASS counts the frames using SR flex counters that exceeded the maximum SDU (MTU) size per stream.
- A count of frames that did not exceed the maximum SDU size filter:
  - The existing bit 4, SR_FLOW_MTU_ERROR is used to count the frames using SR flex counters that did not exceed the maximum SDU (MTU) size per stream.
- A count of frames that were discarded as a result of the operation of the flow meter:
  - A new bit called TAF_FLOW_METER_DROP counts the frames using SR flex counters that were discarded as a result of the operation of the flow meter for that stream.

Chapter 7: Buffer Management Mechanisms

The BCM56070 device supports the following buffer management mechanisms:

- Ingress backpressure (IBP)
- Head-of-line blocking prevention
- Congestion bits

The aim of these schemes is to impose constraints on buffer utilization and to define predictable behavior when encountering congestion events. These constraints may be based upon either the set of counters associated with ingress port utilization or on the set of counters associated with egress port and CoS use. The counters track the buffer utilization in terms of either cells or packets.

7.1 Ingress Backpressure

The goals of the IBP mechanism are to:

- Relieve a congested port.
- Attempt to maintain loss-less behavior.
- Manage buffer resources fairly across ingress ports.

The aim of the IBP mechanism is to support fair access to the buffer resources while also enabling lossless operation across a network of Ethernet switching devices. The IBP mechanism has been designed to be usable for both local IBP support, as well as remote IBP support (end-to-end IBP or end-to-end flow control).

IBP operates by making use of the ingress accounting counters. These counters track the number of cells and packets utilized on a per-ingress-port basis. Based on these counters, PAUSE flow control is used to stop traffic from arriving on ingress ports that have used more than their fair share of buffering resources. The PAUSE flow control frames are generated based on the counter state and a variety of thresholds that are described in this section. By stopping traffic from an offending ingress port, the congestion is relieved.

Each ingress port tracks whether it is in a backpressure state through an ingress counter status bit. This bit is triggered when incoming packets can no longer be stored in the ingress buffer, due to the port is being congested. When packets exceeded their limit threshold, the port then enters IBP state. When the ingress port is in a backpressure state, PAUSE flow control frames with a timer value of (0xFFFF) are periodically sent out of that ingress port. When the ingress port is no longer in the backpressure state, a PAUSE flow control frame with a timer value of 0x00 is sent out and traffic is allowed to flow again. The timer value can be set through a 16-bit field.

The backpressure state associated with each ingress port is based on the state of both the cell and packet counters for that ingress port. An OR function is applied to each of the counters relative to their respective thresholds. If the ingress port is in a backpressure state based on either the cell or packet counters, the ingress port backpressure state is set. Both counters must be in a cleared state for the backpressure state of the ingress port to be cleared.

7.2 Head-of-Line Blocking Prevention

Multiple ports or differential traffic flows that send packets to the same port can create congestion in some architectures. This may lead to the source ports dropping packets destined to other ports, resulting in HOL blocking.

The BCM56070 switch implements two mechanisms, cell-based HOL blocking prevention and packet-based blocking prevention, to prevent HOL blocking on a per-CoS basis for each port. Both mechanisms are simultaneously active. Cell-based HOL blocking prevention is programmable and is based on the total packet memory used by each CoS per port. Packet-based HOL blocking prevention is programmable and is based on the number of packets per CoS queue for each port. Each port can have a maximum of 1024 packets for GbE and FE ports from all the CoS queues waiting in the outgoing queue.

The goals of the head-of-line (HOL) blocking prevention mechanism are as follows:

- Support lossy buffer management
- Manage buffer resources at an egress port with CoS granularity

The aim of this HOL scheme is to support fair access to buffering resources while also optimizing throughput in the system. In contrast, the IBP scheme aimed to support lossless behavior at the possible expense of throughput. The reason for this trade-off is that the IBP thresholds are set to be somewhat conservative to achieve lossless behavior. In addition, PAUSE flow control can generally lead to reduced overall system throughput depending on the round-trip delay between neighboring devices. Alternatively, the HOL mechanism relies on packet dropping to manage the buffering resources. By relying on packet dropping, the HOL thresholds can be set to be more aggressive in terms of buffer utilization thus potentially improving overall system throughput.

The egress buffer usage on a per egress port and CoS basis. Based on these counters and the thresholds described below, decisions are made to drop any newly arriving packets on the ingress ports destined to a particular oversubscribed egress port or CoS queue. The HOL scheme is configurable and operates independently on every CoS queue in the device. Eight CoS queues are associated with every egress port. The HOL feature is applied across all ports. A set of thresholds are associated with each CoS queue. These thresholds provide packet and cell thresholds that define whether a specific queue is in a HOL status state and whether newly arriving packets to an ingress port are dropped.

To handle the reception of in-flight packets when making use of the HOL scheme, a skidpad buffer is provided. The number of in-flight packets can be configured using a skid mark value The actual HOL is computed by taking the specified limit and subtracting the SKIDMARKER value. Consequently, it is important to set the PKTSETLIMIT to be larger than the minimum skid mark setting of four packets.

The HOL mechanism makes use of counters that track cell buffer usage. The cell buffer counter and its associated thresholds aim to provide dynamic memory allocation of the CBP.

7.2.1 Static Memory Allocation

When the device is not in dynamic memory mode, the allocation of cell buffer resources is considered to be statically allocated.

7.2.2 Dynamic Memory Allocation

When the BCM56070 device is in dynamic memory mode, the CoS queues are given a guaranteed number of cell buffers, but they may also exceed that amount based upon a set of dynamic memory thresholds. The dynamic pool of memory is shared among all egress ports and CoS queues. A set of dynamic memory thresholds affect how the pool is distributed among egress ports.

Chapter 8: HiGig Interface

This chapter describes the HiGig interface, which is a Broadcom proprietary stacking protocol and interface. The terms HiGig and HG are used as a generic reference to the HiGig2 protocol.

8.1 HiGig Overview

BCM56070 HiGig support includes the following:

- HiGig2 (HiGig-Lite is not supported)
- Speeds ≥ 10G
- Up to 256 module IDs
- Up to 128 ports per module
- Eight CoS queues per HiGig port

HiGig ports do not support Flexport™ capabilities.

8.2 HiGig Stacking

The HiGig2 stacking protocols are Broadcom-proprietary interconnect protocols. In HiGig mode, the port does not run standard Gigabit Ethernet. A HiGig port can be used only to interconnect other StrataXGS-based devices. HiGig2 allows for extending a greater variety of features across multiple modules. These features include mirroring, trunking, VLAN membership, and congestion awareness. The stacking support is a critical feature for seamlessly supporting a large number of users.

The HiGig2 protocols define a module header used by the devices to properly forward packets across modules and communicate other information to extend the StrataXGS features across multiple devices. Up to 256 modules can be interconnected, not including fabric devices. Any number of fabric devices can be used, as they are not assigned a module ID.

This header is at the front of the Ethernet payload and contains the module header (MH) fields. The VLAN tag (4 bytes) is extracted from the packet and the VLAN information (VID, CFI, and PRIORITY) is inserted into specific fields in the module header. The normal packet FCS field (32-bit CRC) is regenerated to cover the module header.

HiGig2 has a 16-byte module header. A HiGig2 interface must be run at a slightly faster rate due to the overhead of the additional module header bytes. To support 12 Gb/s of front-panel traffic, the interface should be configured to 13 Gb/s.

In scenarios where multiple HiGig ports connect to another StrataXGS-based device, HiGig trunking can distribute traffic across the connected links.

NOTE: In channelized applications, the backplane connection between the BCM56070 and BCM56470 is Ethernet.

8.3 HiGig Stacking

The BCM56070 includes uplink and stacking ports that can support various speeds. The HiGig2 stacking protocols are supported at speeds of ≥ 10G.

Each uplink port is configurable as a standard Ethernet, or HiGig2 port. Uplink ports configured as HiGig links support speeds at the effective switching bandwidth. The device does not support HiGig at speeds less than 10 Gb/s.

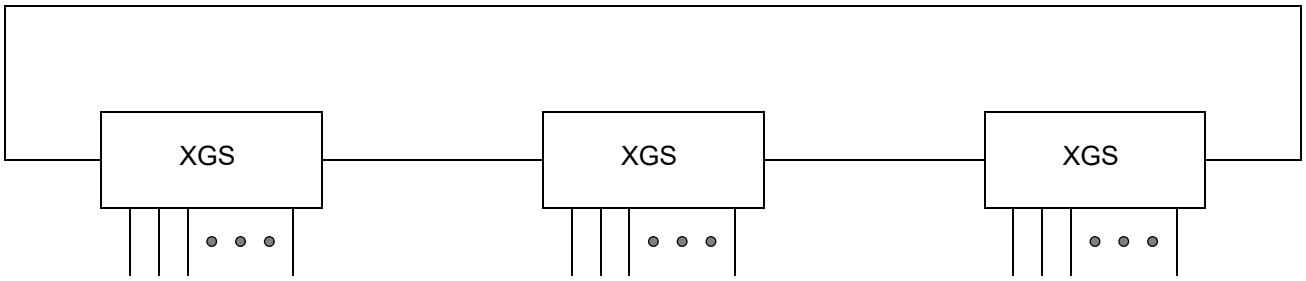
The stacking mechanism provides a number of powerful features across the stack such as trunking, port mirroring, VLAN membership, and a single IP address for management. This capability can enable multiple devices connected together to appear as a single-switch system.

The HiGig protocol provides a mechanism to interconnect multiple modules to form a single-switch system. The system can be several stacked switches or a chassis-based system where several switch and fabric blades can provide high port density. Each packet that is sent across the HiGig link is prepended with a HiGig module header. For HiGig2, this module header is 16 bytes. Therefore, a 64-byte untagged packet that is received on a front-panel 1GbE port will become an 80-byte packet as it is sent across a HiGig link. The HiGig header contains information of the packet type (known unicast, L2 multicast, IPMC, broadcast), source modid port, internal priority, VLAN information, mirroring flags, and other information that is carried to other modules in the system. The module header information is protected by the packets CRC field, which is recalculated to cover the module header.

8.3.1 BroadStack

An inexpensive method for scaling a system is to use a stacking ring. In these systems, the fixed stacking link bandwidth may become a resource limitation as more stacking elements are added. This may be the case under fully meshed traffic patterns. As the number of stacking elements increases, the transit traffic bandwidth requirements also increases. Transit traffic is defined as traffic arriving to a switch device from one stacking link and exiting the same switch device through a different stacking link. In cost-effective systems, these stacking links are typically oversubscribed.

Figure 88: Stacking Configuration Example



Traffic transmitted on to the stacking links is categorized as one of the following two traffic types:

- Add-in traffic – Front-panel port traffic that must be delivered to the stacking links.
- Transit traffic – Traffic that occurs between stacking links.

Because transit traffic may be required to traverse several stacking links, fairness may be an issue in traditional stacking configurations, as add-in traffic and transit traffic compete for bandwidth among the same priority queues.

BroadStack™ technology helps ensure a level of service to transit traffic by having additional queues on the stacking links to allow add-in traffic and transit traffic to be arbitrated as two virtual queues.

8.4 HiGig Trunking

HiGig trunking provides a means of bundling multiple HiGig configured ports together. The BCM56070 supports up to eight trunk groups, with up to eight members per trunk group. Overlapping HiGig trunk groups is not allowed, and dynamic load balancing (DLB) is not supported on HG trunks

All HiGig trunk members must reside within the same module. A HiGig trunk with members from multiple modules is not supported.

Chapter 9: Traffic Management

The BCM56070 device supports the following buffer management mechanisms:

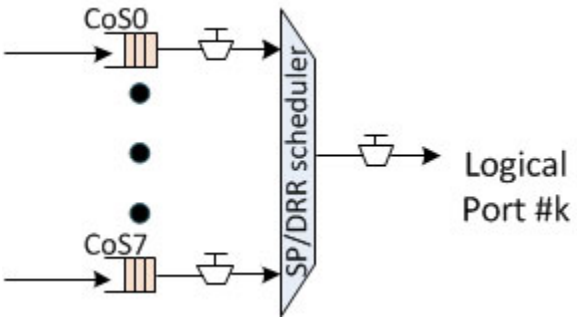
- Differentiated Services QoS
- Single-Rate Three-Color Marker (SrTCM) and Two-Rate Three-Color Marker (TrTCM)
- PAUSE Metering
- CPU Port Scheduler

9.1 Differentiated Services QoS

9.1.1 Legacy Scheduler

The BCM56070 switch supports the legacy 8X1 scheduler as shown in Figure 89. All 56 front-panel logical ports implement an 8X1 scheduler with eight priority queues. Each 8X1 scheduler can be linked to any SGMII_4L, Merlin physical port, or Falcon physical port. Each of the queues has its own queue-based thresholds and counters. Additionally, a PFC can pause one or more individual queues.

Figure 89: Legacy 8X1 Scheduler



The IEEE 802.1D specification defines eight levels of priorities (0 through 7), with priority 7 being the highest priority. This information is carried in the 3-bit priority field of the VLAN tag header and applies to all ports.

The BCM56070 switch supports up to eight CoS queues per egress port. For tagged packets, the incoming packet priority can be mapped to one of the eight CoS queues, based on the priority field in the tag header or from the result of filtering mechanisms. For untagged packets, the CoS priority is derived either from a programmable field within the ARL (VLAN Address tables) or from the result of filtering mechanisms. After the packets are mapped into a CoS queue, they are forwarded or conditioned using one of these schedulers (SP, WRR, and WRR + SP). In addition, the BCM56070 supports guaranteed minimum and maximum bandwidth per CoS.

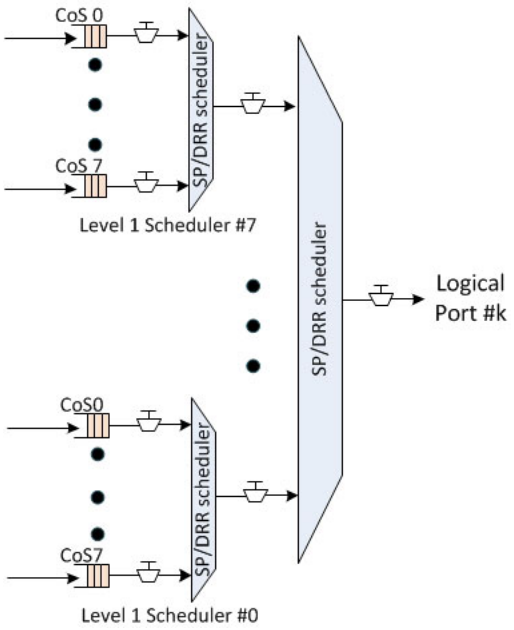
Example: Each CoS can have a minimum bandwidth assigned value of 64 Kb/s and maximum of 1 Gb/s.

9.1.2 Two-Stage Scheduler

The BCM56070 switch supports eight logical ports with a two-stage 8X8 scheduler. There are eight queues per Level 1 scheduler and eight Level 1 schedulers per logical port as shown in Figure 90. The characteristics and guidelines are as follows:

- Any logical port with an 8X8 scheduler can be linked to any physical port.
- Each of the queues has its own queue-based thresholds and counters.
- Each of the Level 1 schedulers has new aggregate buffer management thresholds and counters.
- PFC and PAUSE
  - Mode 0: A PFC pauses one or more individual queues
  - Mode 1: PFC pauses one or more Level 1 Schedulers (not individual queues)

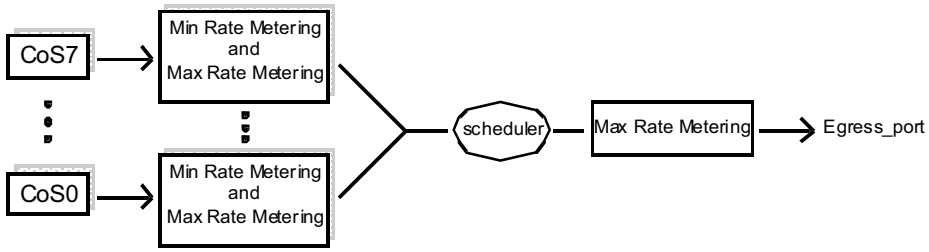
Figure 90: Two-Stage 8X8 Scheduler



9.1.3 Traffic Metering and Shaping

Traffic shaping is supported across each egress port as well as across each of the eight CoS queues associated with every egress port. The traffic shaping aspect is tightly integrated with the scheduler. In fact, the scheduling behavior is configured through the traffic shaping mechanisms and with a set of control masks that modify how the scheduler makes use of the traffic shaping mechanisms. Maximum bandwidth meters are used to feed state information to the scheduler, which responds by modifying its service order across the CoS queues. Minimum bandwidth traffic metering is also provided and operates across each of the eight CoS queues associated with every egress port. Supporting minimum bandwidth metering and maximum bandwidth shaping on a per-CoS queue basis enables system vendors to implement a QoS model across their system. This allows users to configure their CoS queues to support an explicit minimum and maximum bandwidth guarantee by using metering mechanisms that monitor traffic flow on a per CoS basis. It also allows users to provide state information about whether a coarse-grained CoS-flow is above or below the specified minimum and maximum bandwidth specifications. This state information is fed into the scheduler where it then modifies its behavior.

Figure 91: Traffic Metering, Shaping, and Scheduling Overview



9.1.3.1 Minimum Bandwidth CoS Queue Metering

A minimum bandwidth metering mechanism is implemented on a per CoS queue basis at every egress port. The purpose of the minimum bandwidth meter is to enable system vendors to provide strict minimum bandwidth guarantees on a per CoS granularity. This minimum bandwidth meter is implemented using a simple leaky bucket mechanism that tracks whether a CoS queue has consumed its minimum bandwidth. This state information is used by the scheduler to dictate its behavior.

The range of the minimum bandwidth setting for each CoS queue is between 64 Kb/s to 16 Gb/s, in 64-Kb/s increments.

The leaky bucket mechanism works by having a configurable number of tokens leaking out of the bucket at a configurable rate (with a floor at 0). As packets arrive, a number of tokens in proportion to the size of the packet is added to the bucket.

9.1.3.2 Maximum Bandwidth CoS Queue Metering

A maximum bandwidth metering mechanism is implemented on a per CoS queue basis at every egress port. The purpose of the maximum bandwidth meter is to enable system vendors to provide strict maximum bandwidth limitations on a per CoS granularity. This maximum bandwidth meter is implemented using a simple leaky bucket mechanism that tracks whether a CoS queue has consumed its maximum bandwidth. This state information is used by the scheduler to dictate its behavior.

The range of the maximum bandwidth setting for each CoS queue is between 64 Kb/s to 16 Gb/s, in 64-Kb/s increments (same as for minimum bandwidth metering).

9.1.4 Scheduler Modes

The scheduling modes interact with the minimum and maximum bandwidth CoS queue and maximum bandwidth egress port shaping specifications. Each egress port may be configured to have a specific scheduling mode (XQCOSARBSEL register). There are five possibilities: strict priority across CoS queues, round-robin across CoS queues, weighted round-robin, weighted deficit round-robin, and strict priority + WRR or DRR.

9.1.4.1 Strict Priority Across CoS Queues

The strict priority scheduler mode provides strict priority access to the egress port across the CoS queues from highest CoS queue index to the lowest. The purpose of the strict priority scheduler is to provide lower latency service to the higher CoS classes of traffic.

9.1.4.2 Round-Robin Across CoS Queues

The round-robin scheduler mode provides round-robin arbitration across the CoS queues. The scheduler visits each backlogged CoS queue, servicing a single packet at each queue before moving on to the next one. The purpose of the round-robin scheduler is to provide fair access to the egress port bandwidth (at a packet level). This works best when packet sizes are approximately comparable. Minimum and maximum bandwidth metering may also be applied to influence scheduling behavior.

9.1.4.3 Weighted Round-Robin

The WRR scheduler provides a weighted packet round-robin scheme across the CoS queues. The purpose of WRR is to provide a weighted access to the egress port bandwidth (at a packet level). This works best when the average packet size of each coarse-grained CoS flow can be known.

In this algorithm, the scheduler will select one packet from each CoS queue and go around all active queues that have a pending packet. All active CoS queues are programmed with weights according to the required packet distribution. The unit of weight is one packet that can represent anywhere from 64 bytes to 9216 bytes (jumbo frames).

All CoS queues are given credits that are equal to the weight. The round-robin packet selection starts with the highest CoS and attempts to schedule a packet all the way to the lowest CoS. A packet will be chosen from a CoS queue if its weight is non-zero. As each packet is transmitted, one credit for that CoS is decremented. If the weight for any given CoS is non-zero and there is no pending packet, then the credit for that CoS is decremented. The round-robin continues until all CoS queues have zero credits. When the scheduler has reached this point, then the credits are all refreshed according to the weights.

Example: If the weights for cos[7:0] = 8, 7, 6, 5, 4, 3, 2, 1 ..., then the packets are transmitted in round-robin order, so the packets would egress as follows:

- 1st round: cos7, cos6, cos5, cos4, cos3, cos2, cos1, cos0
- 2nd round: cos7, cos6, cos5, cos4, cos3, cos2, cos1
- 3rd round: cos7, cos6, cos5, cos4, cos3, cos2
- 4th round: cos7, cos6, cos5, cos4, cos3
- 5th round: cos7, cos6, cos5, cos4
- 6th round: cos7, cos6, cos5
- 7th round: cos7, cos6
- 8th round: cos7

The 0 value is a special-case setting, which is used to designate that a particular queue be considered as part of a hybrid Strict Priority + WRR configuration. The values 1 to 127 are used to indicate that between 1 and 127 back-to-back packets are to be serviced when the scheduler is servicing a particular CoS queue. If the weight setting is *N* but if there are < *N* packets in the queue, the scheduler continues working and moves on to the next backlogged queue.

9.1.4.4 Weighted Deficit Round-Robin

An inherent limitation of the WRR mode is that bandwidth is allocated in terms of packets. WRR works well if the average packet size for each coarse-grained CoS queue flow is known. However, in most instances this attribute is traffic-dependent and can vary over time. The deficit round-robin (DRR) mode is aimed at addressing this issue. DRR provides a bandwidth allocation scheduler mode that takes into account the variably sized packet issue by maintaining sufficient state information when arbitrating across the CoS queues.

The goal of DRR is to provide coarse-grained flow isolation and bandwidth sharing when arbitrating access to a link among contending CoS flows. This is accomplished by using a modified form of round-robin service. A set of queues is serviced by the DRR scheduler, where each queue is associated with a particular CoS. These queues are serviced in round-robin order while taking into account two state variables: the quantum and the credit counter. Each CoS queue has a configurable quantum associated with it, similar to a WRR weight value. However, the unit for the quantum is in bytes. The purpose of the credit counter is to track the overuse of bandwidth by a particular CoS queue relative to its specified quantum.

DRR operates by servicing the set of backlogged queues in packet round-robin order. Initially, each queue sets its credit counter to its associated (and configurable) quantum value. Every time a packet from a CoS queue is sent, the size of the packet is subtracted from the corresponding credit counter. When the credit counter drops below 0, the queue is no longer serviced until its credits are replenished. All queues are serviced until either they are empty or their credit counter is negative. When this occurs, the credits are replenished. When the credits are replenished, a quantum of credits are added to each CoS queue credit counter. The quantum for each CoS queue may differ based on the configuration. A ceiling operation is applied to the credit counter so that it may not be larger than its specified quantum. This is required in the case where a queue becomes empty before it was able to bring its credit counter below 0.

The credit counter in the DRR mechanism is allowed to drop below 0. This deficit is then tracked so that in the subsequent scheduling rounds, the queue is given fewer credits. By tracking this deficit, DRR enables byte accurate bandwidth sharing despite the fact that traffic contains variable length packets.

Another important attribute of the BCM56070 devices, DRR implementation is how the credit counter is replenished and how the credit counter is updated when the queue becomes empty. In the BCM56070 devices implementation, a queue is not reset to zero if the queue becomes empty while being serviced.

9.2 Single-Rate Three-Color Marker and Two-Rate Three-Color Marker

Metering is a ContentAware metering engine feature used to monitor and control traffic bandwidth. There is a traffic bandwidth profile associated with each flow. The packet is considered in-profile if it conforms to the bandwidth profile and is considered out-of-profile if it does not conform to the bandwidth profile. Metering is supported even with jumbo packets.

The BCM56070 supports both the two-rate three-color marker (TrTCM) scheme and the single-rate three-color marker (SrTCM) scheme, compliant with RFC 2697 and RFC 2698, respectively. Metering is implemented by a dual token bucket structure. In each bucket, a token represents availability of bandwidth. The size of the bucket represents the available burst size for this meter and the refresh token count indicates the committed information rate (CIR). There is a meter associated with every rule in the ContentAware policy engine, where the meter ID is equivalent to the Flow ID for the packet matched in the ContentAware processor. The bandwidth rate control is programmable in steps of 64 Kb/s up to line rate, on both the GbE port and the 10GbE ports.

Chapter 10: Traffic Conditioning

The BCM56070 device supports the following traffic conditioning mechanisms:

- Priority Flow Control
- Weighted Random Early Detection (WRED)

10.1 Priority Flow Control

Priority Flow Control (PFC) is a standard compliant backpressure mechanism implemented in the BCM56070. It is implemented as a priority aware version of the Link Level flow control mechanism. The goal of PFC is to backpressure congested priority traffic flow without affecting the traffic flows of uncongested priorities. PFC can be used in a network with real-time or time-sensitive traffic because of its capability to provide differential treatment to Traffic Classes. For example, using PFC lower priority Internet traffic can be backpressured leaving the higher priority traffic like VoIP and streaming video flowing through the link.

A PFC frame is defined to have similar properties as a standard MAC control frame, except that the opcode is changed to 01-01. The MAC DA, EtherType, and opcode fields are user-programmable in the device to give flexibility for future standard enhancement.

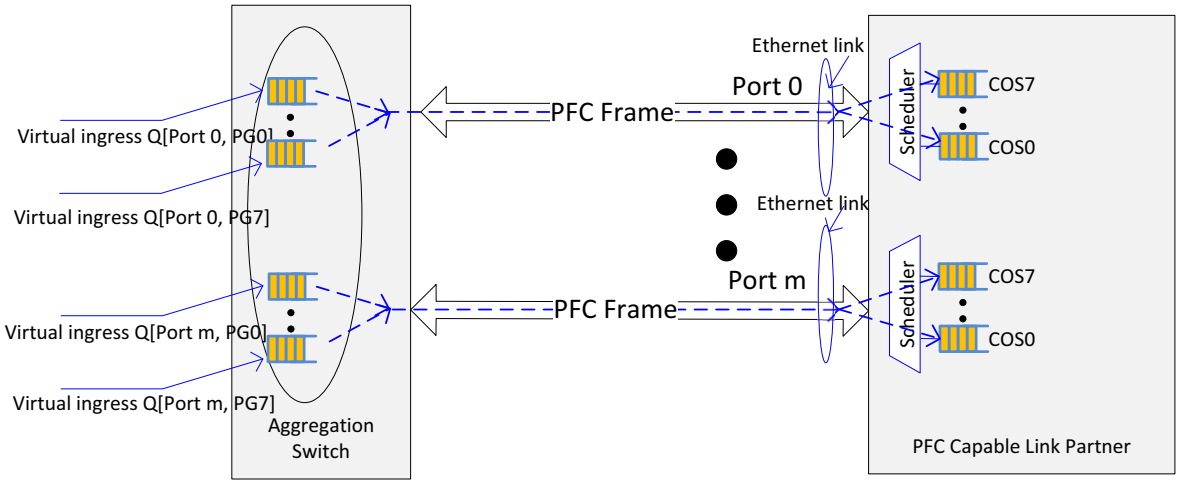
Table 10: PFC Frame Format

Length (Octets)	Field	Description
6	MAC DA	Destination MAC address
6	MAC SA	Source MAC address
2	EtherType	Type field = 88-08
2	MAC Control OPCODE	MAC control frame opcode = 01-01
2	Priority Enable Vector	Priority Enable Vector: Each bit corresponds to a priority. The BCM56070 supports eight priority, therefore, most significant octet in the field is set to zero. If the bit value == 1, then the corresponding timer value is valid.
2	Time (0)	The pause timer value for priority vector[n]. The units are slot times.
2	Time (1)	See the previous description for Time (0).
2	Time (2)	
2	Time (3)	
2	Time (4)	
2	Time (5)	
2	Time (6)	
2	Time (7)	

Figure 92 illustrates PFC enabled Ethernet links. On one end of a PFC enabled Ethernet link in the diagram there is a BCM56070 and on the other end is a PFC-enabled link partner. The BCM56070 maps the eight priority levels in IEEE 802.1P to eight Priority Groups (PGs). MMU maintains a packet and a cell counter to track buffer congestion in a virtual ingress queue for PG corresponding to priority of the packet received on the port. Buffer congestion is caused by packets received at a specific priority level on a PFC enabled link and stored in the switch. Once the number of cells or packets cross respective configured threshold for the Priority Group, the BCM56070 generates a PFC frame and transmits it to the link

partner when the link becomes available. After the link partner receives a PFC message with a priority enabled in the priority vector and with non-zero timer value, the scheduler for the link in the link partner stops transmitting only the frames with the congested priority. It resumes transmission after the timer expires or receives another PFC frame with 0 for the congested priority group timer value, indicating no more buffer congestion in the BCM56070.

Figure 92: PFC Initiator Function



The goals of the priority flow control (PFC) mechanism are as follows:

- Relieve a congested priority on a port.
- Maintain a loss-less behavior on the congested priority.
- Maintain low latency on high priority traffic.
- Manage buffer resources fairly across ingress ports.

PFC is implemented in addition to IBP on all network ports of the BCM56070 irrespective of link speed. This means PFC is supported in all I/O modes (1G, 2.5G, 10G, RXAUI, XAUI, and RXAUI-5) on Ethernet links configured to carry native Ethernet packets. However, if a network port is configured to support HiGig2, SAFC is supported on the link instead of PFC leveraging the same implementation with a different flow control frame format.

10.1.1 PFC Generation and Reception

10.1.1.1 PFC Reception

Once a PFC frame is received on a port, the Priority Enable vector in the PFC frame (see Figure 92) is extracted along with the respective time values. The vector is mapped to one or more CoS queues based on the PRI2COS_PFC configurations. If a Priority Enable vector corresponding to a CoS queue is set and the time value is non-zero, transmission of frames from that priority transmit queue is halted by the scheduler at the packet boundary and a timer is loaded with the time value for the corresponding class. The time value is in units of 512-bit times and the time, in ns, depends on the interface speed. No packet is transmitted from the transmit queue until either the timer value expires or another PFC frame is received.

If another PFC frame is received with the bit for the class is enabled in the Priority Enable vector and time value is 0, packet transmission from that priority queue is resumed. Otherwise, if the class is enabled in the Priority Enable vector and time value is non-zero, packet scheduling from that transmit queue is halted again and the timer is loaded with the time value in the new PFC frame.

By default, the MAC passes PFC frames, similar to a standard IEEE 802.3x Pause control frame to the switch and the switch drops the PAUSE or PFC frames. Optionally, the PFC frame received on a port P can be forwarded to another port of the switch or the CPU by overriding the default drop action of the switch through using IFP actions.

10.2 Weighted Random Early Detection

10.2.1 WRED Overview

Random early detection (RED) is an active queue management mechanism that aims to improve overall Internet performance. Queue management is defined as a mechanism that manages the length of packet queues in a device by dropping packets when needed. RED aims to discard or mark packets randomly and preemptively before buffer space is exhausted. TCP protocol relies on dropped packets as a feedback mechanism for controlling the packet rates by a TCP host. Packet dropping in RED is done probabilistically so as to avert TCP global synchronization. By preemptively dropping packets probabilistically, overall network throughput can be better optimized. If the incoming packet is Explicit Congestion Notification (ECN) enabled and the destined queue is also ECN enabled, then the RED mechanism marks the 2-bit ECN field in the IP header of a TCP packet as ‘b11 instead of dropping it. The effect of ECN on the TCP host is similar to the one if RED had dropped the packet. The TCP host uses ECN marking as a congestion indication in the network and slows its transmission rate until congestion is cleared.

RED operates by having an averaging statistic used to track each queue length. This is an exponentially weighted moving average (EWMA). Based on this averaging statistic, the egress queue manager probabilistically drops or marks packets based on a profile defined for this queue. The drop or mark profile defines what the probability is given a specific average queue size. RED provides active queue management on a single queue associated with an egress port. Weighted random early detection (WRED) is the same RED mechanism, except there are separate RED profiles on a per CoS and packet color basis. The probability of packet dropping or marking from a queue in WRED is a function of the average queue size (AQS), as well as a number of parameters including the probability curves.

Chapter 11: Network Monitoring

This chapter covers the network monitoring features supported by the BCM56070.

11.1 BroadShield

The device supports the following BroadShield™ features and operations:

- Port security
- Hardware protocol checker
- sFlow traffic monitoring

11.1.1 Port Security

The device supports the following features on a per-port-basis for security applications:

- Port blocking
- MAC blocking
- Per-flow blocking
- Broadcast and multicast DLF packet control
- IEEE 802.1aa-compatible

Per-port blocking can be used to prevent certain ports from sending packets to other ports, even if they are on the same VLAN. Each ingress port can be individually prevented from sending packets to any set of egress ports. This is set under EGRESS_MASK register table.

MAC address blocking can be used to prevent packets with an identified source MAC address from being forwarded to specific ports. Up to 32 MAC addresses can be controlled in this fashion.

The ContentAware processor has an action to block packets from being transmitted on given ports. This feature can be used to implement per flow blocking.

Each ingress port can prevent broadcast, unknown multicast, and DLF packets from being forwarded to specific ports. In addition to port security applications, this feature can be used to prevent unwanted packets from being forwarded to certain ports in the VLAN (including the CPU port, where DLF traffic may not be desirable).

11.1.2 sFlow Traffic Monitoring

sFlow is a statistical sampling (count-based or packet-based) technology for monitoring traffic in data networks containing switches and routers. On average, one packet in N is sampled for analysis. An element of randomness is introduced into the sampling process to prevent synchronization with any periodic patterns in the traffic. Sampling does provide a result that can accurately characterize the error. It provides the ability to continuously monitor application level traffic flows at wire speed on all interfaces simultaneously.

Packet sampling is typically performed by the switching or routing ASICs, providing wire speed performance. The state of the forwarding or routing table entries associated with each sampled packet is also recorded. The device provides sFlow support by providing support for time-based packet sampling that is programmable by the software.

Sampling is accomplished using a random number generator to sample the ingress and egress traffics. Each port has a 24-bit random number which is used to compare against the threshold setting. The decision of copying packets to CPU for datagram analysis is based on the comparison between threshold settings and the random numbers.

In addition to standard Management Information Base (MIB) counters such as remote monitoring (RMON), RMON MIB extensions for switched networks (SMON), and SNMP, the device also supports RFC 3176 for sFlow traffic monitoring. sFlow uses a random sampling scheme to characterize traffic flows. Due to randomness, on average, one in every N-packets is captured and analyzed. Although this type of sampling does not provide 100% accuracy, it does provide a result with quantifiable accuracy. The traffic monitoring consists of an sFlow agent and sFlow analyzer. The device has a built-in sFlow agent. It is up to application layer to implement the sFlow analyzer. The sFlow agent within the device uses time-based randomness to sample traffic flows. The sampled data is then forwarded to the sFlow analyzer in the original packet format with an sFlow reason code.

The device supports sFlow sampling for ingress and egress paths.

11.2 Network Management Support

The BCM56070 provides a set of counters to support the following MIB specifications:

- RMON statistics group (IETF RFC 2819)
- SNMP interface group (IETF RFC 1213 and RFC 2863)
- Ethernet-like MIB (IETF RFC 1643)
- Ethernet MIB (IEEE 802.3u)
- Bridge MIB (IETF RFC 1493)

11.3 Mirroring

Mirroring is useful for monitoring ingress or egress traffic on specific ports. The mirrored traffic can be sent to an unused port, where a network sniffer can connect and monitor the traffic. When a port is ingress mirrored, any packet received on that specific port is sent to the sniffer port. When a port is egress mirrored, any packet transmitted from that specific port is sent to the sniffer port. The StrataXGS terminology refers to the sniffer port as the mirror-to-port (MTP).

The BCM56070 supports the XGS®-III style mirroring functions with the following enhancements:

- Each packet may be ingress mirrored up to two copies.
- Each packet may be egress mirrored up to two copies.
- Total mirrored copies are up to four packets.

The following are the traditional XGS-III style mirroring features:

- Mirror packets destined for a specified egress port (egress mirroring).
- Egress mirroring of packets sent by the CPU.
- Mirror packets ingressing a specified port (ingress mirroring).
- Mirror packets on a specific ingress-egress pair.
- Mirror packets of a specific source MAC, destination MAC address pair (conversation).
- Mirroring across stack.
- Remote Switch Port Analyzer (RSPAN).
- Encapsulated Remote Switch Port Analyzer (ERSPAN).

The device supports ingress and egress mirroring both locally and across the HiGig ports. For HiGig ingress, there are two cases to support in regards to mirroring:

- Mirror only. In this case, the packet is only mirrored. No address learning or lookups are performed.
- Switch and mirror. In this case, the switch destination and the mirror-to-port reside in the same module. Therefore, the packet is switched as well as mirrored. Normal address learning is performed.

There are four MTP ports supported for ingress mirroring and there are four MTPs supported for egress mirroring. The MTP port can be a single port or a trunk group. Up to eight physical ports can be assigned to a single trunk group.

The following summarizes mirroring support:

- Number of Mirror ports:
  - Four MTPs for ingress mirrored packets
  - Four MTPs for egress mirrored packets
  - Ingress MTPs and egress MTPs can be same or different
  - Total number of MTPs supported per chip: eight
- An MTP port can be a logical port (trunk).
- For a mirrored packet, no VLAN membership check. The MTP port does not need to be member of all VLANs.
- Ingress mirrored packets are sent unmodified (as the packet ingressed).
- Egress mirrored packets are sent modified with a VLAN tag (always tagged).
- Both switching and mirroring is allowed on a single port (switch and MTP). Both the switched and mirror copy are received.
- If a packet is ingress and egress mirrored, two copies of the packet are sent to the MTP ports.
  - Unmodified packet to ingress MTP port
  - Modified packet (if any) to egress MTP port

11.3.1 Ingress Mirroring

Ingress mirroring is supported for the following configurations:

- Ingress mirroring to local MTP
- Ingress mirroring to remote MTP
- MTP is local trunk
- MTP is a trunk group spread across modules

11.3.2 Egress Mirroring

Users can assign an MTP based on an egress port that the packet is sent from. Egress mirroring is supported for the following configurations:

- Egress mirroring to local MTP.
- Egress mirroring to remote MTP.
- MTP is a local trunk.
- MTP is a trunk group spread across modules.

11.3.3 Trunk Mirroring

The device supports mirroring across a trunk group. Up to eight physical ports can be assigned to a trunk.

11.3.4 MAC-Based Mirroring

The device supports MAC-based mirroring for the following configurations:

- Egress mirroring to local MTP.
- Egress mirroring to remote MTP.
- MTP is local trunk.
- MTP is a trunk group spanning multiple modules.

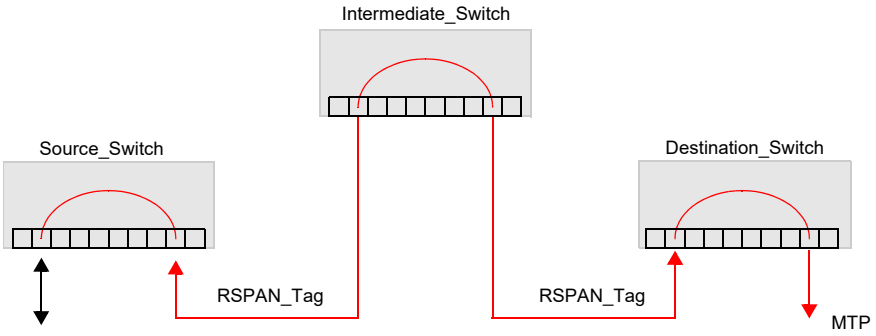
11.3.5 Flow-Based Mirroring

The BCM56070 supports ingress and egress mirroring through the ContentAware Processor (ICAP).

11.3.6 RSPAN Mirroring

Remote Switched Port Analyzer (RSPAN) is also known as remote port mirroring. The device supports remote mirroring by adding a RSPAN VLAN tag to the packet as it egresses the local the mirror-to-port (MTP). RSPAN allows users to assign a MTP in the system that is different than the local MTP on the switch. The following figure illustrates the use of RSPAN under the following network connections.

Figure 93: RSPAN Network Connections



11.3.7 Encapsulated Remote Port Analyzer

Encapsulated Remote Port Analyzer (ERSPAN) allows for mirroring collection points to be located anywhere across a routed network. This is achieved by encapsulating the L2 mirrored packet using GRE with IP delivery. After the packet has been encapsulated, the packet can be forwarded throughout the L3 routed network.

The data section contains the full L2 mirrored packet, except any IEEE 802.1Q tags. The total header added to the mirrored packet is 38 bytes to 42 bytes. This includes the MAC header, IP header, and GRE header sections. The CRC will be recalculated to cover all headers and the L2 mirrored packet payload.

Figure 94: ERSPAN Encapsulation

Figure 95: ERSPAN MAC Header

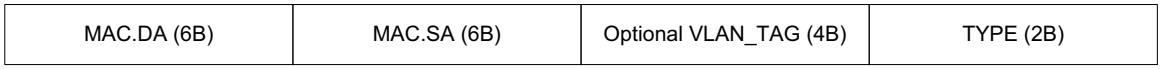
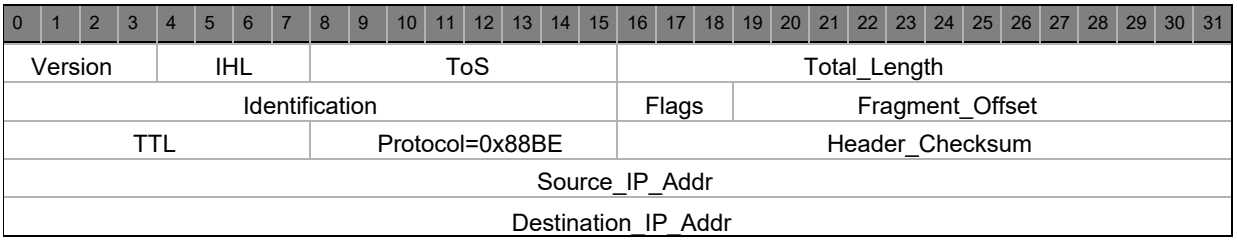


Figure 96: ERSPAN IP Header



The do not fragment bit should be set in the IP header for ERSPAN packets because ERSPAN destinations may be unable to perform the reassembly of fragments.

Figure 97: ERSPAN GRE Header

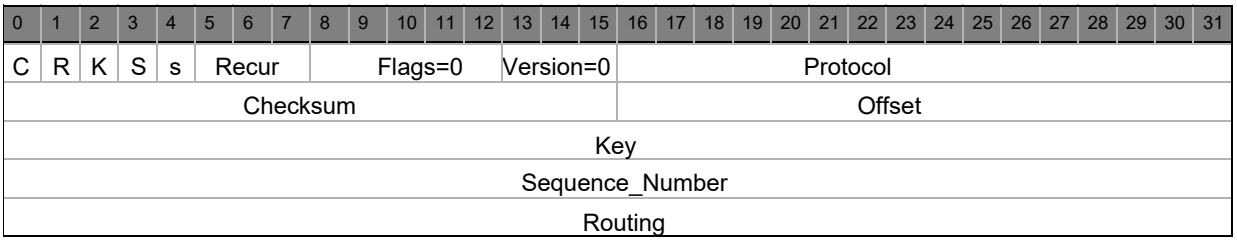
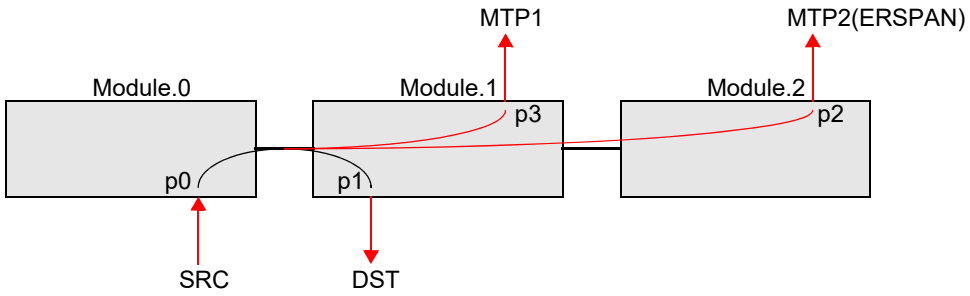


Table 11: ERSPAN Header Bit Definitions

Bit Name	BIT_WIDTH	Description
C	1	Checksum present: 0: Not present 1: Checksum field is present and contains valid information
R	1	Routing present: 0: Not present 1: Offset field is present and contains valid information
K	1	Key present: 0: Not present 1: Key field is present and contains valid information
S	1	Sequence number present: 0: Not present 1: Sequence number is present and contains valid information
s	1	Strict source route. This bit should only be set if all of the routing information consists of strict source routes.
Recur	3	Recursion control. Contains the number of additional encapsulations that are allowed.
Flags	5	Flags. Reserved, must be 0.
Version	3	GRE protocol version, must be 0.
Protocol	16	Protocol field. Contains the protocol type of the payload packet.
Checksum	16	Checksum. Contains the IP checksum of the GRE header and payload packet. Optional.
Offset	16	Offset. Indicates the byte offset from the start of the Routing field to the first byte of the active source route entry to be examined. Optional.
Key	32	Key. Contains a number which was inserted by the encapsulator. It may be used by the receiver to authenticate the source of the packet. Optional.
Sequence Number	32	Sequence number. Contains a number which is inserted by the encapsulator. It may be used by the receiver to establish the order in which packets have been transmitted from the encapsulator to the receiver. Optional.
Routing	Variable	Routing. Contains a list of SREs. Optional.

Figure 98: Ingress Mirror



A packet ingresses port 0 (p0) on Module.0 and is destined for port 1 (p1) on Module.1. Ingress mirroring is enabled with two mirror-to-ports (MTPs). MTP1 on Module.1 is configured as a normal MTP. MTP2 on Module.2 is configured for ERSPAN. A mirrored packet egressing this port will be encapsulated with an ERSPAN header.

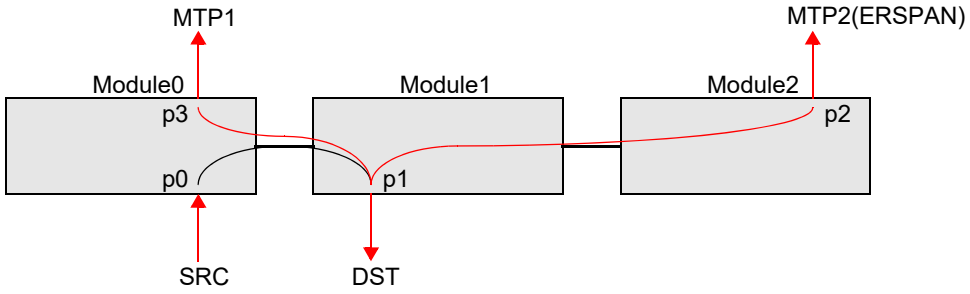
In this example, Module.0 will create the following three copies of the packet:

- Switched copy
- MTP1 mirror copy
- MTP2 mirror copy (ERSPAN encapsulated)

Any intermediate device in the mirror path will merely forward the packet as normal and no packet modification will occur.

In addition, Module.0 will perform a MTU check on the ERSPAN encapsulated packet to ensure that it does not violate any packet size restrictions.

Figure 99: Egress Mirror



A packet ingresses port 0 (p0) on Module.0 and is destined for port 1 (p1) on Module.1. The port 1 Module.1 traffic is egress mirrored. MTP1 on Module.0 is configured as a normal MTP. MTP2 on Module.2 is configured for ERSPAN. A mirrored packet egressing this port will be encapsulated with an ERSPAN header.

In this example, Module.1 will create two copies of the packet: an MTP1 mirror copy and a MTP2 mirror copy that are ERSPAN encapsulated.

The mirror packet destined for MTP1 will be processed normally for an egress mirror packet and then forwarded to Module.0. The mirror packet destined for MTP2 will also be processed normally for an egress mirror packet, but will also be ERSPAN encapsulated. This encapsulation takes place after the normal egress mirror packet processing and will then be forwarded to Module.2.

The mirror destination modules will forward the packet as normal, and no packet modification will occur.

IPv6 is not supported as an ERSPAN delivery method. Only IPv4 is supported.

11.3.8 HiGig Mirroring

The device also supports ingress and egress mirroring across HiGig ports on a remote module.

11.4 Ethernet OAM

11.4.1 Overview

The BCM56070 supports Ethernet Operation, Administration, and Management (OAM). Ethernet OAM allows the service provider to detect, localize, and repair the problems between its domain and the customer. The two main mechanisms within Ethernet OAM that allow such provisions are IEEE 802.1ag and ITU-T Y.1731.

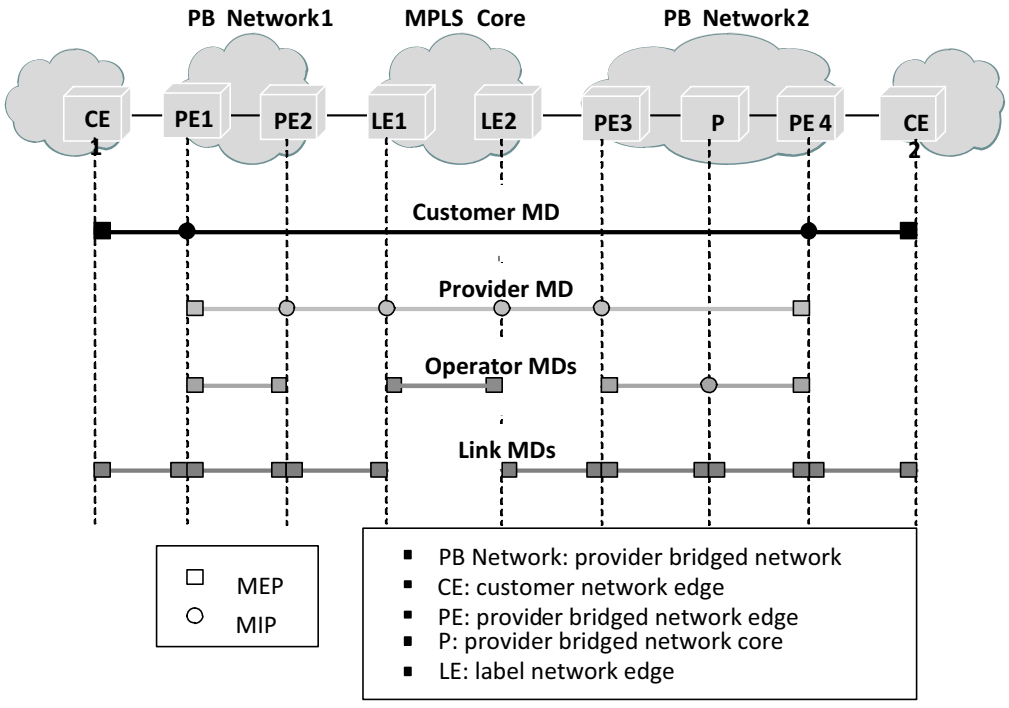
IEEE 802.1ag is also known as Connectivity Fault Management (CFM). IEEE 802.1ag allows the service provider to detect a fault condition on a serviced VLAN between the service provider and the customer. Therefore, IEEE 802.1ag operates on a per VLAN basis. This allows the service provider to detect and isolate the failures.

The ITU-T Y.1731 standard leverages the Hierarchical Maintenance Domain Model defined in IEEE 802.1ag and defines additional performance monitoring services such as Delay Measurement (DM) and Loss Measurement (LM).

This document describes how CFM, DM, and LM are supported in the BCM56070 family.

The following figure shows a flat view of a network for which Ethernet OAM can be deployed.

Figure 100: Ethernet OAM (Flat View)



Ethernet OAM is subdivided into levels of nested Maintenance Domains (MDs) in order to provide hierarchy in an inherently flat transport. Each MD represents a portion of the end-to-end network for which OAM functions will be performed. Operator MDs represent organizations that own and maintain a network of bridges and routers within a certain locality. Provider MDs are comprised of multiple Operator domains to form Wide Area Networks (WANs). Customer MDs are realized with multiple Provider networks to build the end-to-end services they require. Link MDs may be implemented optionally throughout each hop of the entire network. Each domain is represented by an MD Level. The MD Level is carried in a small field in the OAM messages throughout the network. Maintenance association End Points (MEPs) are configured at the boundaries of each MD and are used to initiate and process OAM messages. Maintenance domain Intermediate Points (MIPs) are configured at points internal to the domains and are used only to respond to certain OAM messages (that is, LinkTrace Messages). A group of MEPs within the same MD that provides connectivity fault management for a particular service is called a Maintenance Association (MA). All MEPs within an MA share the same MA Identifier (MAID) and MD Level.

NOTE: Multiple MEPs and MIPs may be configured at a single point in the network in order to provide management visibility to multiple MDs. In this document, Maintenance Point (MP) refers to either a MIP or a MEP.

11.4.2 Connectivity Fault Management

Connectivity Fault Management (CFM) is composed of a suite of protocols including Continuity Check (CC), Loopback Message (LB), and Link Trace Message (LT). The degree to which the support for these are supported using hardware varies. In all cases, the hardware can facilitate protocol packet filtering and forwarding to the host processor. Table 12 is a summary of hardware processing for different CFM packets.

Table 12: CFM Processing

Packet Type	Packet Processing
CCM	Transmission – The device transmits CCMs, periodically, for all MEPs configured on each local bridge port. Reception – The device maintains the CCM database per MEP, per MA and indicates a CFM fault when any of the fault conditions is encountered.
LBM	Transmission – The host CPU initiates the LBM. Reception – The device terminates the LBM and initiates a LBR response packet.
LBR	Transmission – Initiated through hardware in response to a received unicast LBM packet; otherwise, the host CPU initiates the packet. Reception – hardware detects the packet and traps to the CPU.
LTM, LTR	Transmission – The host CPU initiates the packet. Reception – Hardware detects the packet and traps to the CPU.

11.4.2.1 CCM Transmission

The BCM56070 can be configured to transmit Connectivity Check Message (CCM) packets at regular intervals for each locally configured MEP. The interval is adjustable on a per MEP basis and may range from 3.33 ms to 10 minutes. The following figure shows the CCM transmit processing flow. The device injects the CCM into the Ingress Pipeline and uses unused CPU slots. If the CPU slots are full, the transmit CCM is not injected into the pipeline. The transmit CCM is encapsulated with a configurable HiGig2 Module Header to capture the necessary BCM56070 information needed in the packet processing flow. The MA tables that are common to both the transmit and receive CCM processing flows are highlighted in red.

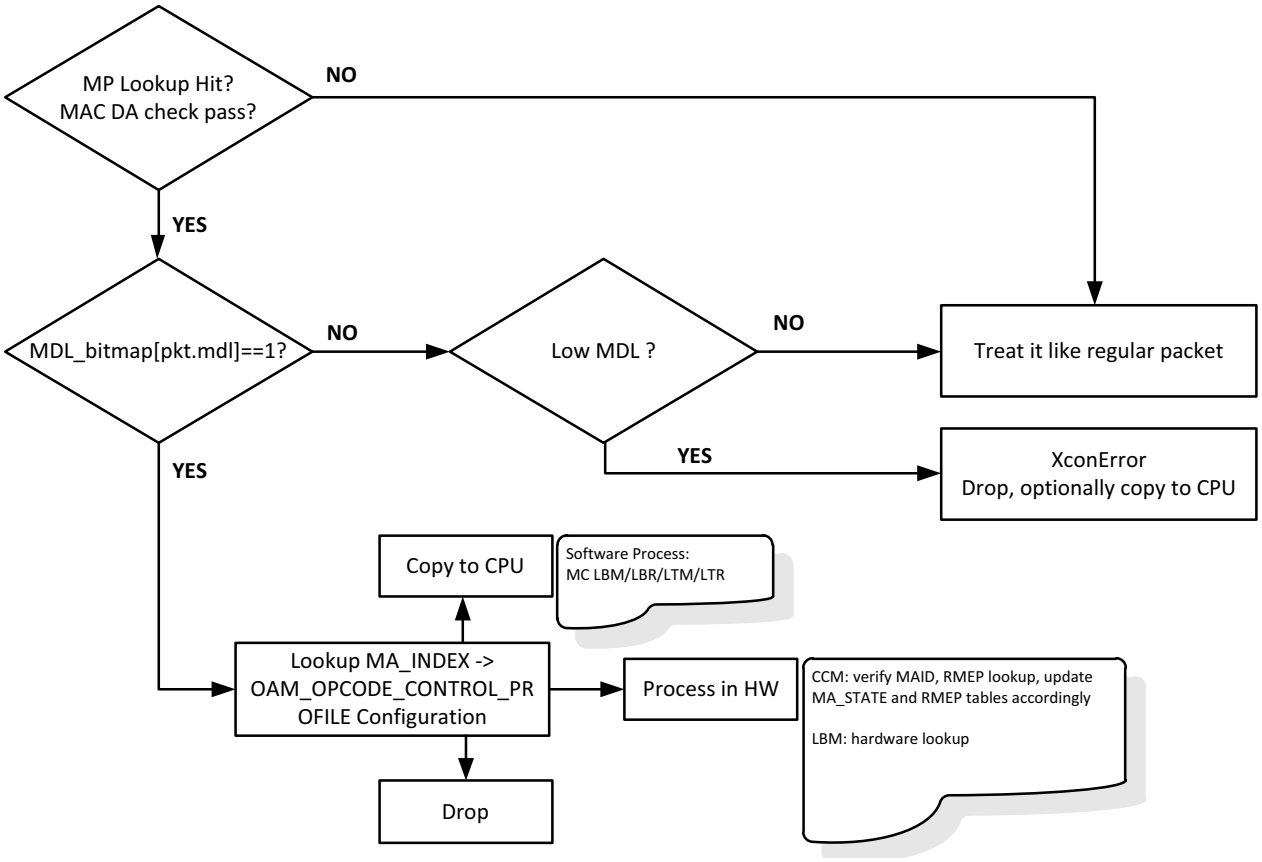
11.4.2.2 CFM PDU Reception

Upon reception of a CFM PDU, such as a CCM message, the first step in processing is to decide whether to bridge, drop, or terminate and process the packet. This involves identifying the LMEP that the CFM PDU is received on and then comparing MD levels to determine how to process the packet. The steps are as follows:

1. MP Identification
2. MDL Check

This process flow is shown in the following figure.

Figure 101: CFM Packet Flow



Because LMEPs are associated with a specific bridge port, VID, and MDL, a {Port, VID} lookup is performed to retrieve an 8-bit MDL bitmap that indicates on which of the eight hierarchy levels an LMEP is configured. The results are as follows:

1. If the bitmap is 0, or the received MDL is greater than any MDL configured on the received {Port, VID}, the packet is bridged normally.
2. If the received MDL is equal to one of the configured MDLs on the {Port, VID}, the packet is terminated and processed.
3. If the first and second conditions are not met, the packet is dropped and the xconError condition is asserted because it was received on an invalid MDL.

11.4.2.2.1 MP Identification

The MP identification involves two elements: MACDA check and MP lookup. MP Identification is passed when both elements are successful. If either misses, the packet is forwarded as a normal data packet.

11.4.2.2.2 Tag Status Check

For port-based MP, do not include a tag status check.

11.4.2.2.3 MDL Check

MDL check involves comparing the MDL level and determining if the packet is to be dropped, forward normally, or OAM processed.

MP lookup is performed to retrieve an 8-bit MDL bitmap that indicates on which of the eight hierarchy levels an LMEP is configured. If the bitmap is 0, or the received MDL is greater than any MDL configured on the received {Port, VID}, the packet is bridged normally. If the received MDL is equal to one of the configured MDLs on the {Port, VID}, the packet is terminated and processed. Otherwise, the packet is dropped and optional copied to CPU.

11.4.2.3 CCM Reception Fault Maintenance

With the MA and RMEP pointers retrieved from the lookups in the previous section, the various fault states can now be checked and updated. The device supports maintenance of IEEE 802.1ag fault states inside the hardware.

11.4.2.4 Per MEP Fault Variables

The following fault states must be maintained on a per MEP basis. The fault states are listed in order of priority from least severe to most severe as described in the IEEE 802.1ag specification. The first three are an aggregation of the faults described in the RMEP Faults section.

someRDIdefect: Logical OR of all *rMEPlastRDI* indications that are part of the same MA as this LMEP.

This is implemented using a per MEP counter in the MA_STATE table called the someRDIdefect_counter. Whenever a CCM is received from an RMEP associated with this LMEP, the following actions occur:

- A transition from 0 to 1 increments the someRDIdefect_counter.
- A transition from 1 to 0 decrements the someRDIdefect_counter.

A non-zero value for this counter indicates that someRDIdefect must be asserted.

someMACStatusDefect: Logical AND of all *rMEPportStatusDefect* OR-ed with logical OR of all *rMEPinterfaceStatusDefect* indications that are part of the same MA as this LMEP.

This defect is not explicitly maintained in hardware. A change in Port or Interface status from any RMEP triggers an interrupt to software. It is then the CPU’s responsibility to maintain the someMACStatusDefect state and to update the software programmable SW_RDI used in outgoing CCMs.

someRMEPCCMdefect: Logical OR of all *rMEPCCMdefect* indications that are part of the same MA as this LMEP.

This is implemented via a per MEP counter in the MA_STATE table, called the someRMEPCCMdefect_counter. Whenever a CCM is received from an RMEP associated with this LMEP, the following actions occur:

- A transition of rMEPCCMdefect from 0 to 1 increments the someRMEPCCMdefect_counter.
- A transition of rMEPCCMdefect from 1 to 0 decrements the someRMEPCCMdefect_counter.

A non-zero value for this counter indicates that someRMEPCCMdefect must be asserted.

errorCCMdefect: A CCM with an invalid MAID was received on this MEP, but one of the following conditions occurred:

- CCM.MEPID not configured in database (RMEP lookup miss).
- CCM.MEPID == LMEP.MEPID (loop detected).
- CCM Interval mismatch.

The transition of this fault state from 0 to 1 starts the following process:

1. The timer is started at 3.5x of the *received* CCM interval when an errorCCMdefect is detected.
2. The defect indication is cleared when the timer expires.
3. The defect indication is reset to 3.5x of the maximum received CCM interval and the current CCM interval if another errorCCM is received on this LMEP.

xconCCMdefect: A CCM with an invalid MAID was received on this LMEP, or the received MDL value was lower than any MDL configured on the bridge port, which indicates a service mismerge condition. The transition of this fault state from 0 to 1 starts the following process:

1. The timer started at 3.5x of the received CCM interval when an xconCCMdefect is detected.
2. The defect indication is cleared when the timer expires.
3. The defect indication reset to 3.5x of the maximum received CCM interval and the current CCM interval if another xconCCM was received on this LMEP.

11.4.2.5 Per MA Fault State

The per-MA fault state is maintained in the MA state table, which is direct-indexed by the MA_PTR from the LMEP and RMEP tables described previously. A separate MA aging process initiates the timer function for each MA in this table to timeout the xconCCM and errorCCM conditions. The timestamps of the last errorCCM and xconCCM are noted for each MA, and are compared to 3.5x of the CCM Interval of the last errorCCM or xconCCM during the aging process. If the difference exceeds the bounds defined by the CCM Interval, the appropriate fault condition is deasserted. A current and sticky fault state is maintained for each error condition. The sticky fault state tracks the first transition from 0 to 1, and is cleared by software. The current fault state set and clear behavior is described in the following table.

IEEE 802.1ag requires CCM messages with certain types of faults to be stored as part of the state machine variables. These packets are copied to the CPU as follows:

- CCMs that triggered a DefErrorCCM fault – This fault occurs when the MEPID in the received CCM caused an RMEP lookup failure, or it matches the MEPID of the receiving MEP or the CCM Interval field in the received CCM does not match what is configured for the receiving MEP.
- CCMs that triggered a DefXconCCM fault – This fault occurs when CCM was received on this LMEP with invalid MAID or received MDL value is lower than any MDL configured on the bridge port.

This mechanism is supported through a chip-wide configuration register.

11.4.2.6 Loopback Reply

Loopback Reply (LBR) is implemented in hardware to offload the host CPU and to deter DoS attacks using CFM LBM messages. The LBM processing is composed of two parts:

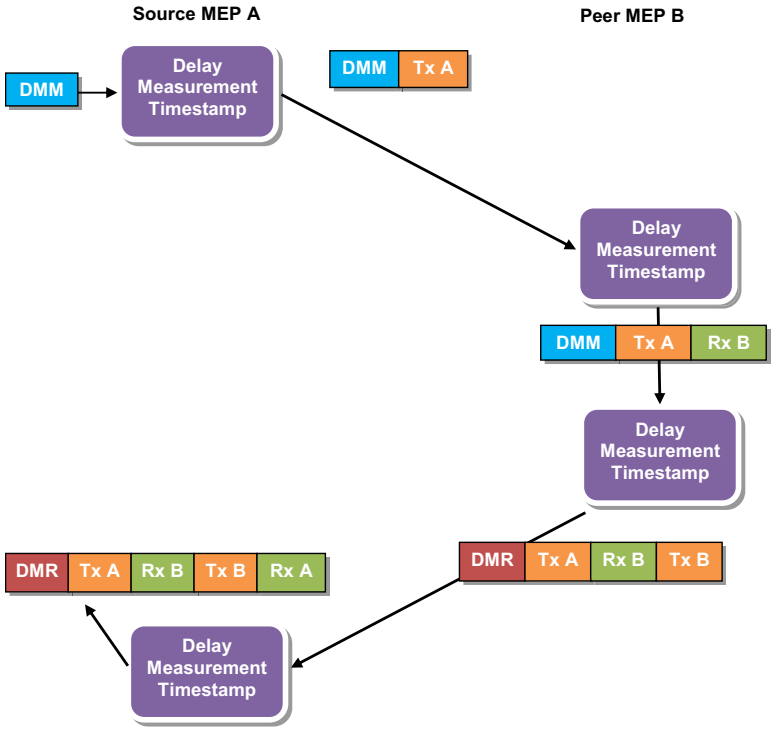
1. Verify that the destination LMEP is local. Lookup is performed on packets {VID, MACDA} to identify whether the destination MEP is local.
2. Respond to the LBM with an LBR as shown in the following table.

11.4.3 Delay Measurement

The device supports Y.1731 delay measurement by performing a two-way four-timestamp delay measurement. See Figure 102 for the four-step process, which includes the following actions:

1. MEP A initiates a Delay Measurement Message (DMM). It inserts the local timestamp, DMM.TxA, and transmits the DMM.
2. MEP B receives the DMM from MEP A and retrieves the local timestamp, DMM.RxA.
3. MEP B initiates a Delay Measurement Response (DMR) message. It copies DMM.TxA and DMM.RxA into the DMR message. It also inserts current local timestamp DMM.TxB, and transmits the DMR.
4. MEP A receives the DMR from MEP B and retrieves the local timestamp, DMR.RxB.

Figure 102: Delay Measurement



The delay is calculated using the four timestamps.

Sampled 2-way delay = (DMR.RxB – DMM.TxA) – (DMR.TxB – DMM.RxA)

11.4.3.1 DMM and DMR TX Packet Flow

DMM and DMR packets are initiated by the CPU. The CPU is responsible for inserting the correct SGLP or SVP, CoS value, Packet Color, and MDL Level. The CPU injects this packet into the ingress pipeline. This packet will then need to match on an OAM rule in the IFP. The IFP will need to do a lookup based on {SVP} or {SGLP, OVID}. On a hit, the following bits will be retrieved from the FP_Policy Table: OAM_LMEP_EN, OAM_DM_EN, OAM_LMEP_MDL, and OAM_TX.

Further processing occurs if the following FP Policy Table conditions are met:

- OAM_LMEP_EN =1
- OAM_DM_EN =1
- OAM_TX = 1

If so, the hardware will check that the following packet conditions are met:

- Pkt.EtherType = OAM
- Pkt.OPCODE = DMM or DMR
- Pkt.OAM MDL = FP_POLICY.OAM_LMEP_MDL

If all of the preceding conditions exist, then the timestamp is sampled from the 64-bit free running counter and inserted into the packet during egress. The clock is sampled prior to be being queued into memory.

11.4.3.2 DMM and DMR RX Packet Flow

DMM and DMR packets are received on the front-panel ports. The packets must match a rule in the IFP with lookup based on {DVP} or {DGLP, OVID}. On a hit, the following bits will be retrieved from the FP_Policy Table: OAM_LMEP_EN, OAM_DM_EN, OAM_LMEP_MDL, and the OAM_TX bits.

Further processing occurs if the following FP Policy Table conditions are met:

- OAM_LMEP_EN =1
- OAM_DM_EN =1
- OAM_TX = 0

If so, the hardware will check that the following packet conditions are met:

- Pkt.EtherType = OAM
- Pkt.OPCODE = DMM or DMR
- Pkt.OAM MDL = FP. OAM_LMEP_MDL

If all of the preceding conditions exist, then the timestamp is sampled from the 64-bit free running counter and copied to the CPU along with the incoming packet. The clock is sampled prior to be being queued into memory.

For the packet copied to the CPU, the CPU OPCODE will have bit 48, OAM_LMDM_OFFSET, set to 1 and the timestamp will be located in the TIMESTAMP and TIMESTAMP_UPPER fields of the DMA descriptor.

Chapter 12: ContentAware Engine

The ContentAware processors (CAPs) are designed to provide support for ACL, DSCP, and QoS type of applications. Filtering and metering can be performed on traffic originating from the Ethernet ports or the HiGig ports.

- **VCAP** – VLAN-based ContentAware Processor. For packets ingressing on the GbE and 10GbE ports. Preingress lookup for L2 and L3 pre-routed packets. The VCAP can be used to assign VLAN, based on payload content or assign class ID for the IFP to perform additional filtering and thus provide a mechanism for hierarchical filtering.
- **ICAP** – Ingress ContentAware processor. For packets ingressing on the GbE and 10 GbE ports. Ingress lookups occur on L2 and L3 pre-routed packets. The ICAP is the most flexible and powerful of the three CAPS. Filtering can done by parsing the first 128 bytes of the packet using either predefined protocol fields such as VLAN, L2, and L3 addresses or using User Defined fields. Assigning a new priority, route, drop, or redirecting the packet are some of the actions that can be performed.
- **ECAP** – Egress ContentAware processor. The egress lookups are for packet egressing the GbE and 10GbE ports. The ECAP is for post MMU packet modification. The inner or outer VLAN ID, drop packets, or meter the packets can be changed based on parsing criteria.


BROADCOM®