7.10.2.2.4 MSI and MSI-X Capabilities

Both MSI and MSI-X are implemented in the 82599. MSI-X vectors can be assigned per VF. MSI is not supported for the VFs.

See Section 9.3.8.1 for more details of the MSI-X and PBA tables implementation.

7.10.2.2.5 VPD Capability

VPD is implemented only once and is accessible only from the PF.

7.10.2.2.6 Power Management Capability

The 82599 does not support power management per VF. The power management registers exist for each VF, but only the D0 power state is supported.

7.10.2.2.7 Serial ID

The serial ID capability is not supported in VFs.

7.10.2.2.8 Error Reporting Capabilities (Advanced and Legacy)

All the bits in this capability structure are implemented only for the PF. Note that the VMs see an emulated version of this capability structure. See Section 7.10.2.4 for details.

7.10.2.3 FLR Capability

The *FLR* bit is required per VF. Setting of this bit resets only a part of the logic dedicated to the specific VF and does not influence the shared part of the port. This reset should disable the queues, disable interrupts and the stop receive and transmit process per VF.

Setting the PF *FLR* bit resets the entire function.

7.10.2.4 Error Reporting

Error reporting includes legacy error reporting and Advanced Error Reporting (AER) or role-based capability.

The legacy error management includes the following functions:

1. Error capabilities enablement. These are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. This includes:
    a. SERR# Enable
    b. Parity Error Response
    c. Correctable Reporting Enable

   d. Non-Fatal Reporting Enable
   e. Fatal Reporting Enable
   f. UR Reporting Enable
2. Error status in the configuration space. These should be set separately for each VF. This includes:
   a. Master Data Parity Error
   b. Signaled Target Abort
   c. Received Target Abort
   d. Master Abort
   e. SERR# Asserted
   f. Detected Parity Error
   g. Correctable Error Detected
   h. Non-Fatal Error Detected
   i. Unsupported Request Detected

AER capability includes the following functions:

1. Error capabilities enablement. The *Error Mask*, and *Severity* bits are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. These includes:
   a. Uncorrectable Error Mask Register
   b. Uncorrectable Error Severity Register
   c. Correctable Error Mask Register
   d. ECRC Generation Enable
   e. ECRC Check Enable
2. Non-Function Specific Errors Status in the configuration space.
   a. Non-Function Specific Errors are logged in the PF
   b. Error logged in one register only
   c. VI avoids touching all VFs to clear device level errors
   d. The following errors are not function specific
      - All Physical Layer errors
      - All Link Layer errors
      - ECRC Fail
      - UR, when caused by no function claiming a TLP
      - Receiver Overflow
      - Flow Control Protocol Error
      - Malformed TLP
      - Unexpected Completion

3. Function Specific Errors Status in the configuration space.
    a. Allows Per VF error detection and logging
    b. Help with fault isolation
    c. The following errors are function specific
        - Poisoned TLP received
        - Completion Timeout
        - Completer Abort
        - UR, when caused by a function that claims a TLP
        - ACS Violation
4. Error logging. Each VF has it's own header log.
5. Error messages. In order to ease the detection of the source of the error, the error messages should be emitted using the requester ID of the VF in which the error occurred.

7.10.2.5 Alternative Routing ID (ARI) and IOV Capability Structures

In order to allow more than eight functions per end point without requesting an internal switch, as usually needed in virtualization scenarios, the PCI-SIG defines the ARI capability structure. This is a new capability that enables an interpretation of the *Device* and *Function* fields as a single identification of a function within the bus. In addition, a new structure used to support the IOV capabilities reporting and control is defined. Both structures are described in sections Section 9.4.3 and Section 9.4.4. Refer to the following section for details on the Requester ID (RID) allocation to VFs.

7.10.2.6 RID Allocation

RID allocation of the VF is done using the *Offset* field in the IOV structure. This field should be replicated per VF and is used to do the enumeration of the VFs.

Each PF includes an offset to the first associated VF. This pointer is a relative offset to the Bus/Device/Function (BDF) of the first VF. The *Offset* field is added to PF's requester ID to determine the requester ID of the next VF. An additional field in the IOV capability structure describes the distance between two consecutive VF's requester IDs.

7.10.2.6.1 BDF Layout

7.10.2.6.1.1 ARI Mode

ARI allows interpretation of the device ID part of the RID as part of the function ID inside a device. Thus, a single device can span up to 256 functions. In order to ease the decoding, the least significant bit of the function number points to the physical port number. The *Next* bits indicate the VF number. The following table lists the VF RIDs.

The layout of RID's used by the 82599 is reported to the operating system via the PCIe IOV capability structure. See Section 9.4.4.6.

Table 7-70 RID per VF — ARI Mode

Port	VF#	B,D,F	Binary	Notes
0	PF	B,0,0	B,00000,000	PF
1	PF	B,0,1	B,00000,001	PF
0	0	B,16,0	B,10000,000	Offset to first VF from PF is 128.
1	0	B,16,1	B,10000,001	
0	1	B,16,2	B,10000,010	
1	1	B,16,3	B,10000,011	
0	2	B,16,4	B,10000,100	
1	2	B,16,5	B,10000,101	
...				
0	63	B,31,6	B,11111,110	
1	63	B,31,7	B,11111,111	Last

7.10.2.6.1.2 Non-ARI Mode

When ARI is disabled, non-zero devices in the first bus cannot be used, thus a second bus is needed to provide enough RIDs. In this mode, the RID layout is as follows:

Table 7-71 RID per VF — Non-ARI Mode

Port	VF#	B,D,F	Binary	Notes
0	PF	B,0,0	B,00000,000	PF
1	PF	B,0,1	B,00000,001	PF
0	0	B+1,16,0	B+1,10000,000	Offset to first VF from PF is 384.
1	0	**B+1**,16,1	B+1,10000,001	
0	1	**B+1**,16,2	B+1,10000,010	
1	1	**B+1**,16,3	B+1,10000,011	
0	2	**B+1**,16,4	B+1,10000,100	
1	2	**B+1**,16,5	B+1,10000,101	
...				
0	63	**B+1**,31,6	B+1,11111,110	
1	63	**B+1**,31,7	B+1,11111,111	Last

Note: When the device ID of a physical function changes (because of LAN disable or LAN function select settings), the VF device IDs changes accordingly.

7.10.2.7 Hardware Resources Assignment

The main resources to allocate per VM are queues and interrupts. The assignment is static. If a VM requires more resources, it might be allocated to more than one VF. In this case, each VF gets a specific Ethernet MAC address/VLAN tag in order to enable forwarding of incoming traffic. The two VFs are then teamed in software.

7.10.2.7.1 PF Resources

A possible use of the PF is for a configuration setting without transmit and receive capabilities. In this case, it is not allocated to any queues but is allocated to one MSI-X vector.

The PF has access to all the resources of all VMs, but it is not expected to make use of resources allocated to active VFs.

7.10.2.7.2 Assignment of Queues to VF

See Section 7.2.1.2.1 for allocating Tx queues.

See Section 7.1.2.2 for allocating Rx queues.

The following table lists the Tx and Rx queues to VF allocation.

Table 7-72 Queue to VF Allocation

VF	Queues in 16 VMs Mode	Queues in 32 VMs Mode	Queues in 64 VMs Mode
0	0-7	0-3	0-1
1	8-15	4-7	2-3
...	...	...	...
15	120-127	...	...
...		...	...
31		124-127	...
...			...
63			126-127

7.10.2.7.3 Assignment of MSI-X Vector to VF

See Section 7.3.4.3 for allocating MSI-X vectors in IOV mode.

7.10.2.8 CSR Organization

CSRs can be divided into three types:

- Global Configuration registers that should be accessible only to the PF. For example, link control and LED control. These types of registers also include all of the debug features such as the mapping of the packet buffers and is responsible for most of the CSR area requested by the 82599. This includes per VF configuration parameters that can be set by the PF without performance impact.
- Per-VF parameters — For example, per VF reset, interrupt enable, etc. Multiple instances of these parameters are used only in an IOV system and only one instance is needed for non IOV systems.
- Per-queue parameters that should be replicated per queue — For example, head, tail, Rx buffer size, DCA tag, etc. These parameters are used by both a VF in an IOV system and by the PF in a non-IOV mode.

In order to support IOV without distributing the current drivers operation in legacy mode, the following method is used:

- The PF instance of BAR0 continues to contain the legacy and control registers. It is accessible only to the PF. The BAR enables access to all the resources including the VF queues and other VF parameters. However, it is expected that the PF driver does not access these queues in IOV mode.
- The VF instances of BAR0 provide control on the VF specific registers. These BARs have the same mapping as the original BAR0 with the following exceptions:
  a. Fields related to the shared resources are reserved.
  b. The queues assigned to a VF are mapped at the same location as the first same number of queues of the PF.
- Assuming some backward compatibility is needed for IOV drivers, The PF/VF parameters block should contain a partial register set as described in Section 8.3.

7.10.2.9 SR IOV Control

In order to control the IOV operation, the physical driver is provided with a set of registers. These include:

- The mailbox mechanism described in the next section.
- The switch and filtering control registers described in Section 7.10.3.10.
- PFVFLRE register indicating that a VFLR reset occurred in one of the VFs (bitmap).

7.10.2.9.1 VF-to-PF Mailbox

The VF drivers and the PF driver require some means of communication between them. This channel can be used for the PF driver to send status updates to the VFs (such as link change, memory parity error, etc.) or for the VF to send requests to the PF (add to VLAN).

Such a channel can be implemented in software, but requires enablement by the VMM vendors. In order to avoid the need for such an enablement, the 82599 provides such a channel that enables direct communication between the two drivers.

The channel consists of a mailbox. Each driver can then receive an indication (either poll or interrupt) when the other side wrote a message.

Assuming a maximum message size of 64 bytes (one cache line), a memory of 64 bytes x 64 VMs =
4 KB. 512 bytes is provided per port. The RAM is organized as follows:

Table 7-73 Mailbox Memory

RAM Address	Function	PF BAR 0 Mapping[1]	VF BAR 0 Mapping[2]
0 — 63	VF0 <-> PF	0 — 63	VF0 + MBO
64 — 127	VF1 <-> PF	64 — 127	VF1 + MBO
....			
(4 KB-64) — (4 KB-1)	VF63<-> PF	(4 KB-64) — (4 KB-1)	VF63 + MBO

1. Relative to mailbox offset.
2. MBO = mailbox offset in VF CSR space.

In addition for each VF, the VFMailbox and PFMailbox registers are defined in order to coordinate the transmission of the messages. These registers contain a semaphore mechanism to enable coordination of the mailbox usage.

The PF driver can decide which VFs are allowed to interrupt the PF to indicate a mailbox message using the PFMBIMR mask register.

The following flows describe the usage of the mailbox:

Table 7-74 PF-to-VF Messaging Flow

Step	PF Driver	Hardware	VF #n driver
1	Set PFMailbox[n].PFU		
2		Set PFU bit if PFMailbox[n].VFU is cleared	
3	Read PFMailbox [n] and check that PFU bit was set. Otherwise wait and go to step 1.		
4	Write message to relevant location in VMFBMEM.		
5	Set the PFMailbox[n].STS bit and wait for ACK[1].		
6		Indicate an interrupt to VF #n.	
7			Read the message from VFMBMEM.

Table 7-74 PF-to-VF Messaging Flow (Continued)

Step	PF Driver	Hardware	VF #n driver
8			Set the VFMailbox.ACK bit.
9		Indicate an interrupt to PF.	
10	Clear PFMailbox[n].PFU		

1. The PF might implement a timeout mechanism to detect non-responsive VFs.

Table 7-75 VF-to-PF Messaging Flow

Step	PF Driver	Hardware	VF #n Driver
1			Set VFMailbox.VFU.
2		Set VFU bit if VFMailbox[n].PFU is cleared.	
3			Read VFMailbox [n] and check that VFU bit was set. Otherwise wait and go to step 1.
4			Write message to relevant location in VFMBMEM.
5			Set the VFMailbox.REQ bit.
6		Indicate an interrupt to PF.	
7	Read PFMBICR to detect which VF caused the interrupt.		
8	Read the adequate message from VFMBMEM.		
9	Set the PFMailbox.ACK bit.		
10		Indicate an interrupt to VF #n.	
11			Clear VFMailbox.VFU.

The content of the message is hardware independent and is determined by software.

The messages currently assumed by this specification are:

- Registration to VLAN/multicast packet/broadcast packets — A VF can request to be part of a given VLAN or to get some multicast/broadcast traffic.
- Reception of large packet — Each VF should notify the PF driver what is the largest packet size allowed in receive.
- Get global statistics — A VF can request information from the PF driver on the global statistics.
- Filter allocation request — A VF can request allocation of a filter for queuing/ immediate interrupt support.

- Global interrupt indication.
- Indication of errors.

7.10.2.10 DMA

7.10.2.10.1 RID

Each VF is allocated a RID. Each DMA request should use the RID of the VM that requested it. See Section 7.10.2.6 for details.

7.10.2.10.2 Sharing of the DMA Resources

The outstanding requests and completion credits are shared between all the VFs. The tags attached to read requests are assigned the same way as in a non-virtualized setting, although in VF systems tags can be re-used for different RIDs. See Section 3.1.3.1.

7.10.2.10.3 DCA

The DCA enable is common to all the devices (all PFs and VFs). Given a DCA enabled device, each VM might decide for each queue, on which type of traffic (data, headers, Tx descriptors, Rx descriptors) the DCA should be asserted and what is the CPU ID assigned to this queue.

Note: There are no plans to virtualize DCA in the IOH. Thus, the physical CPU ID should be used in the programming of the CPUID field.

7.10.2.11 Timers and Watchdog

7.10.2.11.1 TCP Timer

The TCP timer is available only to the Physical Function (PF). It might indicate an interrupt to the VFs via the mailbox mechanism.

7.10.2.11.2 IEEE 1588

IEEE 1588 is a per-link function and thus is controlled by the PF driver. The VMs have access to the real time clock register.

7.10.2.11.3 Watchdog

The watchdog was originally developed for pass-through NICs where virtualization is not a viable. Thus, this functionality is used only by the PF.

7.10.2.11.4 Free Running Timer

The free running timer is a PF driver resource the VMs can access. This register is read only to all VFs and is reset only by the PCI reset.

7.10.2.12 Power Management and Wake Up

Power management is a PF resource and is not supported per VF.

7.10.2.13 Link Control

The link is a shared resource and as such is controllable only by the PF. This includes interface settings, speed and duplex settings, flow control settings, etc. The flow control packets are sent with the station Ethernet MAC address stored in the EEPROM. The watermarks of the flow control process and the time-out value are also controllable by the PF only. In a DCB environment, the parameters of the per TC flow control are also part of the PF responsibilities.

Linksec is a per-link function and is controlled by the PF driver.

Double VLAN is a network setting and as such should be common to all VFs.

7.10.2.13.1 Special Filtering Options

Pass bad packets is a debug feature. As such, pass bad packets is available only to the PF. Bad packets are passed according to the same filtering rules of the regular packets.

Note: Pass bad packets might cause guest operating systems to get unexpected packets. As a result, it should be used only for debug purposes of the entire system.

Receiving long packets is enabled separately per Rx queue in the RXDCTL registers. As this impacts the flow control thresholds, the PF should be made aware of the decision of all the VMs. Because of this, the setup of TSO packets is centralized by the PF and each VF might request this setting.

7.10.3 Packet Switching

7.10.3.1 Assumptions

The following assumptions are made:

- The required bandwidth for the VM-to-VM loopback traffic is low. That is, the PCIe bandwidth is not congested by the combination of the VM-to-VM and the regular incoming traffic. This case is handled but not optimized for. Unless specified otherwise, Tx and Rx packets should not be dropped or lost due to congestion caused by loopback traffic.

- If the buffer allocated for the VM-to-VM loopback traffic is full, it is acceptable to back pressure the transmit traffic of the same TC. This means that the outgoing traffic might be blocked if the loopback traffic is congested.
- The decision on local traffic is done only according to the Ethernet DA address and the VLAN tag. There is no filtering according to other parameters (IP, L4, etc.). The switch has no learning capabilities. In case of double VLAN mode, the inner VLAN is used for the switching functionality.
- The forwarding decisions are based on the receive filtering programming.
- No packet switching between TCs.
- Coexistence with IPSEC offload: Any loopback VM-to-VM traffic should not use the IPSEC offload (the *IPSEC* bit must be cleared in the advanced Tx data descriptor). IPsec processing of Tx packets destined to a local VM must be handled by software.
- Coexistence with TimeSync: time stamp is not sampled for any VM-to-VM loopback traffic.
- Coexistence with Double VLAN: When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, transmit-to-receive packet switching should not be enabled.

7.10.3.2 Pool Selection

Pool selection is described in the following sections. A packet might be forwarded to a single pool or replicated to multiple pools. Multicast and broadcast packets are cases of replication, as is mirroring.

The following capabilities determine the destination pools of each packet:

- 128 Ethernet MAC address filters (RAH/RAL registers) for both unicast and multicast filtering. These are shared with L2 filtering. For example, the same Ethernet MAC addresses are used to determine if a packet is received by the switch and to determine the forwarding destination.
- 64 shared VLAN filters (PFVLVF and PFVLVFB registers) — each VM can be made a member of each VLAN.
- Hash filtering of unicast and multicast addresses (if the direct filters previously mentioned are not sufficient)
- Forwarding of broadcast packets to multiple pools
- Forwarding by Ethertype
- Mirroring by pool, VLAN, or link

7.10.3.3 Rx Packets Switching

Rx packet switching is the second of three stages that determine the destination of a received packet. The three stages are defined in Section 7.1.2.

As far as switching is concerned, it doesn't matter whether the 82599's virtual environment operates in IOV mode or in Next Generation **VMDq** mode.

When operating in replication mode, broadcast and multicast packets can be forwarded to more than one pool, and is replicated to more than one Rx queue. Replication is enabled by the Rpl_En bit in the PFVTCTL register.

7.10.3.3.1 Replication Mode Enabled

When replication mode is enabled, each broadcast/multicast packet can go to more than one pool. Finding the pool list of any packet is provided in the following steps:

1. Exact unicast or multicast match — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list.
2. Broadcast — If the packet is a broadcast packet, add pools for which their PFVML2FLT.BAM bit (Broadcast Accept Mode) is set.
3. Unicast hash — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add pools for which their PFVML2FLT.ROPE bit (Accept Unicast Hash) is set.
4. Multicast hash — If the packet is a multicast packet and the prior steps yielded no pools, check it against the Multicast Hash Table (MTA). If there is a match, add pools for which their PFVML2FLT.ROMPE bit (Receive Multicast Packet Enable) is set.
5. Multicast promiscuous — If the packet is a multicast packet, take the candidate list from prior steps and add pools for which their PFVML2FLT.MPE bit (Multicast Promiscuous Enable) is set.
6. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.

Note: In a VLAN network, untagged packets are not expected. Such packets received by the switch should be dropped, unless their destination is a virtual port set to receive these packets. The setting is done through the PFVML2FLT.AUPE bit. It is assumed that VMs for which this bit is set are members of a default VLAN and thus only MAC queuing is done on these packets.

7. Default pool — If the pool list is empty at this stage and the PFVTCTL.Dis_Def_Pool bit is cleared, then set the default pool bit in the target pool list (from PFVTCTL.DEF_PL).
8. Ethertype filters — If one of the Ethertype filters (ETQF) is matched by the packet and queuing action is requested and the Pool Enable bit in the ETQF is set, the pool list is set to the pool pointed to by the filter.
9. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.
10. Mirroring — Each of the four mirroring rules adds its destination pool (PFMRCTL.MP) to the pool list if the following applies:
    a. Pool mirroring — PFMRCTL.VPME is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.

b. VLAN port mirroring — PFMRCTL.VLME is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the VMVLAN register.

c. Uplink port mirroring — PFMRCTL.UPME is set, the pool list is not empty.

d. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers. Note that this stage appears twice in order to handle mirroring cases.

7.10.3.3.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode, the pool list of any packet always contains one pool only according to the following steps:

1. Exact unicast or multicast match — If the packet DA matches one of the exact filters (RAL/RAH), use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list.
2. Unicast hash — If the packet is a unicast packet, and the prior step yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add the pool for which their PFVML2FLT.ROPE (Accept Unicast Hash) bit is set. Refer to the software limitations described after step 7.
3. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.
4. Default pool — If the pool list is empty at this stage and the PFVTCTL.Dis_Def_Pool bit is cleared, then set the default pool bit in the target pool list (from PFVTCTL.DEF_PL).
5. Multicast or broadcast — If the packet is a multicast or broadcast packet and was not forwarded in step 1 and 2, set the default pool bit in the pool list (from PFVTCTL.DEF_PL).
6. Ethertype filters — If one of the Ethertype filters (ETQF) is matched by the packet and queuing action is requested and the Pool Enable bit in the ETQF is set, the pool list is set to the pool pointed by the filter.
7. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.

The following software limitations apply when replication is disabled:

- Software must not set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it's guaranteed that the packet is sent to only one queue by other means (such as VLAN).
- Software must not set per-VM promiscuous bits (multicast or broadcast).
- Software must not set the *ROPE* bit in more than one PFVML2FLT register.
- Software should not activate mirroring.

7.10.3.4 Tx Packets Switching

Tx switching is used only in a virtualized environment to serve VM-to-VM traffic. Packets that are destined to one or more local VMs are directed back (loopback) to the Rx packet buffers. Enabling Tx switching is done by setting the PFDTXGSWC.LBE bit. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRTH[n].RTH fields regardless if flow control is activated on the 82599.

Tx switching rules are very similar to Rx switching in a virtualized environment, with the following exceptions:

- If a target pool is not found, the default pool is used only for broadcast and multicast packets.
- A unicast packet that matches an exact filter is not sent to the LAN.
- Broadcast and multicast packets are always sent to the external LAN.
- A packet might not be sent back to the originating pool (even if the destination address is equal to the source address) unless loopback is enabled for that pool by the PFVMTXSW[n] register.

The detailed flow for pool selection as well as the rules that apply to loopback traffic is as follows:

- Loopback is disabled when the network link is disconnected. It is expected (but not required) that system software (including VMs) does not post packets for transmission when the link is disconnected. Note that packets posted by system software for transmission when the link is down are buffered.
- Loopback is disabled when the RXEN (*Receive Enable*) bit is cleared.
- Loopback packets are identified by the *LB* bit in the receive descriptor.

Note: When Tx switching is enabled, the host must avoid sending packets longer than 9.5 KB as this hangs the Tx path.

7.10.3.4.1 Replication Mode Enabled

When replication mode is enabled, the pool list for any packet is determined according to the following steps:

1. Exact unicast or multicast match — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, take the MPSAR[n] bits as a candidate for the pool list.
2. Broadcast — If the packet is a broadcast packet, add pools for which their PFVML2FLT.BAM bit (Broadcast Accept Mode) is set.
3. Unicast hash — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add pools for which their PFVML2FLT.ROPE bit (Accept Unicast Hash) is set.
4. Multicast hash — If the packet is a multicast packet and the prior steps yielded no pools, check it against the Multicast Hash Table (MTA). If there is a match, add pools for which their PFVML2FLT.ROMPE bit (Receive Multicast Packet Enable) is set.
5. Multicast promiscuous — If the packet is a multicast packet, take the candidate list from prior steps and add pools for which their PFVML2FLT.MPE bit (Multicast Promiscuous Enable) is set.

6. Filter source pool — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
7. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.
8. Forwarding to the network — Packets are forwarded to the network in the following cases:
   a. All broadcast and multicast packets.
   b. Unicast packets that do not match any exact filter.
9. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (pre mirroring step). Refer to the notes after step 11.
10. Mirroring — Each of the following three mirroring rules adds its destination pool (PFMRCTL.MP) to the pool list if the following applies:
    a. Pool mirroring — PFMRCTL.VPME is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
    b. VLAN port mirroring — PFMRCTL.VLME is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the VMVLAN register.
    c. Downlink port mirroring — PFMRCTL.DPME is set and the packet is sent to the network.
11. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (post mirroring step). Refer to the following notes.

Note: The PFVFRE filtering is applied only after the decision to forward the packet to network and/or local pool (based on MAC address and VLAN). If a packet that matches an exact MAC address is set to be forwarded to a local pool, it is not sent to the network regardless of the PFVFRE setting. Therefore, when a pool is disabled, software should also clear its exact MAC address filters before clearing the PFVFRE.

7.10.3.4.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode the pool list for any packet always contains one pool only according to the following steps:

1. Exact unicast or multicast match — If the packet DA matches one of the exact filters (RAL/RAH), take the MPSAR[n] bits as a candidate for the pool list.
2. Unicast hash — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add the pool for which their PFVML2FLT.ROPE bit (Accept Unicast Hash) is set. Refer to the software limitations that follow.
3. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — PFVLVFB[n]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.

4. Multicast or broadcast — If the packet is a multicast or broadcast packet and was not forwarded in step 1 and 2, set the default pool bit in the pool list (from PFVTCTL.DEF_PL).
5. Filter source pool — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
6. Forwarding to the network — Packets are forwarded to the network in the following cases:
   a. All broadcast and multicast packets.
   b. Unicast packets that do not match any exact filter.
7. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.

Note: The PFVFRE filtering is applied only after the decision to forward the packet to network and/or local pool (based on MAC address and VLAN). If a packet that matches an exact MAC address is set to be forwarded to a local pool, it is not sent to the network regardless of the PFVFRE setting. Therefore, when a pool is disabled, software should also clear its exact MAC address filters before clearing the PFVFRE.

The following software limitations apply when replication is disabled:

1. It is software's responsibility not to set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it is guaranteed that the packet is sent to only one queue by other means (such as VLAN)
2. Software must not set per-VM promiscuous bits (multicast or broadcast).
3. Software must not set the *ROPE* bit in more than one PFVML2FLT register.
4. Software should not activate mirroring.

7.10.3.5 Mirroring Support

The 82599 supports four separate mirroring rules, each associated with a destination pool (mirroring can be done into up to four pools). Each rule is programmed with one of the four mirroring types:

1. Pool mirroring — reflect all the packets received to a pool from the network.
2. Uplink port mirroring — reflect all the traffic received from the network.
3. Downlink port mirroring — reflect all the traffic transmitted to the network.
4. VLAN mirroring — reflect all the traffic received from the network in a set of given VLANs (either from the network or from local VMs).

Note: Reflecting all the traffic received by any of the pools (either from the network or from local VMs) is supported by enabling mirroring of all pools.

Mirroring and replication on FCoE traffic is not supported on receive if the ETQF filters define FCoE packets and on transmit if the packets are indicated as FCoE (by setting the FCoE bit in the TUCMD field in the Transmit Context Descriptor).

Mirroring modes are controlled by a set of rule control registers:

- PFMRCTL — controls the rules to be applied and the destination port.
- PFMRCTL — controls the VLAN ports as listed in the PFVLVF table taking part in the VLAN mirror rule.
- PFMRVM — controls the pools taking part in the pool mirror rule.

7.10.3.6 Offloads

The general rule is that offloads are executed as configured for the pool and queue associated with the receive packet. Some special cases:

- If a packet is directed to a single pool, then offloads are determined by the pool and queue for that packet.
- If a packet is replicated to more than one pool, then each copy of the packet is offloaded according to the configuration of its pool and queue.
- If replication is disabled, offloads are determined by the unique destination of the packet.

The following subsections describe exceptions to the previously described special cases.

7.10.3.6.1 Local Traffic Offload

The following capabilities are not supported on the loopback path:

- The Ethertype filters do not apply.
- Padding to a legal packet size is not supported.
- The following offload capabilities are only supported if XSUM offload is provided on the Tx path for the packet: RSS, 5-tuple filters, VLAN strip. The reason is that when XSUM is not offloaded, software does not provide the necessary offload offsets with the Tx packet.
- Header split/replication is not supported for NFS.
- Receive Side Coalescing (RSC) is not supported.
- FCoE offloads are not supported.
- IPSec offload is not supported.

7.10.3.6.2 Rx Traffic Offload

- Security offloads (LinkSec, IPsec) are managed globally and not per pool.
- CRC offload is a global policy. CRC strip is enabled or disabled for all received packets.

7.10.3.7 Congestion Control

- Tx packets going through the local switch are stored in the Rx packet buffer, similar to packets received from the network. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRTH[n].RTH fields regardless if flow control is activated on the 82599.

The 82599 guarantees that one TC flow is not affected by congestion in another TC.

Receive and local traffic are provided with the same priority and performance expectations. Packets from the two sources are merged in the Rx packet buffers, which can in general support both streams at full bandwidth. Any congestion further in the pipeline (such as lack of PCIe bandwidth) evenly affects Rx and local traffic.

7.10.3.8 Tx Queue Arbitration and Rate Control

In order to guarantee each pool with adequate bandwidth, a per-pool bandwidth control mechanism is added to the 82599. Each Tx pool gets a percentage of the transmit bandwidth and is guaranteed it can transmit within its allocation. This arbitration is combined with the TC arbitration. See additional details on DCB Tx capabilities in Section 7.7.2.2.

7.10.3.9 Security Features

The 82599 allows some security checks on the inbound and outbound traffic of the switch.

7.10.3.9.1 Inbound Security

Each incoming packet (either from the LAN or from a local VM) is filtered according to the VLAN tag so that packets from one VLAN cannot be received by pools that are not members of that VLAN.

7.10.3.9.2 Outbound Security

MAC anti-spoofing

Each pool is associated with one or more Ethernet MAC addresses on the receive path. The association is determined through the MPSAR registers. The MAC anti-spoofing capability insures that a VM always uses a source Ethernet MAC address on the transmit path that is part of the set of Ethernet MAC addresses defined on the Rx path. A packet with a non-matching SA is dropped, preventing spoofing of the Ethernet MAC address. This feature is enabled in the PFVFSPOOF.MACAS field, and can be enabled per Tx pool.

Note: Anti-spoofing is not available for VMs that hide behind other VMs whose Ethernet MAC addresses are not part of the RAH/RAL Ethernet MAC Address registers. In this case, anti-spoofing should be done by software switching, handling these VMs.

VLAN anti-spoofing

Each pool is associated with one or more VLAN tags on the receive path. The association is determined through the PFVLVF and PFVLVFB registers. The VLAN anti-spoofing capability insures that a VM always uses a VLAN tag on the transmit path that is part of the set of VLAN tags defined on the Rx path. A packet with a non-matching VLAN tag is dropped, preventing spoofing of the VLAN tag. This feature is enabled in the PFVFSPOOF.VLANAS field, and can be enabled per Tx pool.

Note: If VLAN anti-spoofing is enabled, then MAC anti-spoofing must be enabled as well.

When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN anti-spoofing should not be enabled.

VLAN tag validation

In PCI-SIG IOV scenarios the driver might be malicious, and thus may fake a VLAN tag. The 82599 provides the ability to override the VLAN tag inserted by a VM. The possible behaviors are controlled by the PFVMVIR[n] registers as follows:

- Use descriptor value — to be used in case of a trusted VM that can decide which VLAN to send. This option should also be used in case one VM is member of multiple VLANs.
- Always insert default VLAN — this mode should be used for non-trusted or non-VLAN aware VMs. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.
- Never insert VLAN — This mode should be used in a non-VLAN network. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.

Note: The VLAN insertion settings should be done before any of the queues of the VM are enabled.

When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN validation should not be enabled.

7.10.3.10 Switch Control

The PF driver has some control of the switch logic. The following registers are available to the PF for this purpose:

PFVTCTL: - VT Control register — contains the following fields:

- Replication Enable (Rpl_En) — enables replication of multicast and broadcast packets — both in incoming and local traffic. If this bit is cleared, Tx multicast and broadcast packets are sent only to the network and Rx multicast and broadcast packets are sent to the default pool.
- Default Pool (DEF_PL) — defines the target pool for packets that passed L2 filtering but didn't pass any of the pool filters. This field is invalid when the Dis_Def_Pool bit is set.
- Disable Default Pool (Dis_Def_Pool) — disables acceptance of packets that failed all pool filters.

- PFVFRE — Enables/disables reception of packets from the link to a specific VF. Used during initialization of the VF. See Section 4.2.2.2 for more details.
- PFDTXGSWC (LBE) — VMDQ loopback enables switching of Tx traffic to the Rx path for VM-to-VM communication.
- PFVFSPOOF — MAC Anti-spoof Enable (MACAS) — enables filtering of Tx packet for anti-spoof.
- Local Loopback Enable (LLE) — defines whether or not to allow loopback of a packet from a certain pool into itself.
- Queue Drop Enable (PFQDE) register — A register defining global policy for drop enable functionality when no descriptors are available. It lets the PF override the per-queue SRRCTL[n] Drop_En setting. PFQDE should be used in SR-IOV mode as described in Section 4.6.11.3.1.
- PFVML2FLT — Receive Overflow Multicast Packets (ROMPE) — accept multicast hash — Defines whether or not a pool accepts packets that match the multicast MTA table.
- Receive MAC Filters Overflow (ROPE) — accept unicast hash — Defines whether or not a pool accepts packets that match the unicast PFUTA table.
- Broadcast Accept (BAM) — Defines whether or not a pool accepts broadcast packets.
- Multicast Promiscuous (MPE) — Defines whether or not a pool accepts all multicast packets.
- Accept Untagged Packets Enable (AUPE) — Defines whether or not a pool accepts untagged VLAN packets.
- Mirror Control — See Section 7.10.3.5.
- PFVFTE — Enables/disables transmission of packets to the link to a specific VF. Used during initialization of the VF. See Section 4.2.2.2 for more details.
- PFVLVF/PFVLVFB — VLAN queuing table — A set of 64 VLAN entries with an associated bitmap, one bit per pool. Bits are set for each pool that participates in this VLAN.
- Unicast Table Array (PFUTA) — a 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received unicast packet that misses the MAC filters is compared against the PFUTA. If the relevant bit in the PFUTA is set, the packet is routed to all pools for which the *ROPE* bit is set.
- Multicast Table Array (MTA) — a 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received multicast packet that misses the MAC filters is compared against the MTA. If the relevant bit in the MTA is set, the packet is routed to all pools for which the *ROMPE* bit is set.

In addition, the rate-control mechanism is programmed as described in Section 7.7.2.2.

7.10.4 Virtualization of Hardware

This section describes additional features used in both IOV and Next Generation **VMDq** modes.

7.10.4.1 Per-pool Statistics

Part of the statistics are by definition shared and cannot be allocated to a specific VM. For example, CRC error count cannot be allocated to a specific VM, as the destination of such a packet is not known if the CRC is wrong.

All the non-specific statistics are handled by the PF driver in the same way it is done in non-virtualized systems. A VM might request a statistic from the PF driver but might not access it directly.

The conceptual model used to gather statistics in a virtualization context is that each queue pool is considered as a virtual link and the Ethernet link is considered as the uplink of the switch. Thus, any packet sent by a pool is counted in the Tx statistics, even if it was forwarded to another pool internally or was dropped by the MAC for some reason. In the same way, a replicated packet is counted in each of the pools receiving it.

The following statistics are provided per pool:

- Good packet received count
- Good packet transmitted count
- Good octets received count
- Good octets transmitted count
- Multicast packets received count

Note: All the per VF statistics are read only and wrap around after reaching their maximum value.

7.11 Receive Side Coalescing (RSC)

The 82599 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The 82599 does this by coalescing the incoming frames into a single or multiple buffers (descriptors) that share a single accumulated header. This feature is called RSC. Note that the term Large Receive is used to describe a packet construct generated by RSC.

The 82599 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The 82599 can handle up to 32 concurrent flows per LAN port at any given time. Each flow handled by RSC offload has an associated context. The 82599 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

Figure 7-41 shows a top level flow diagram that is used for RSC functionality. The following sections provide a detailed explanation of this flow as well as the memory structures and device settings that support the RSC functionality.

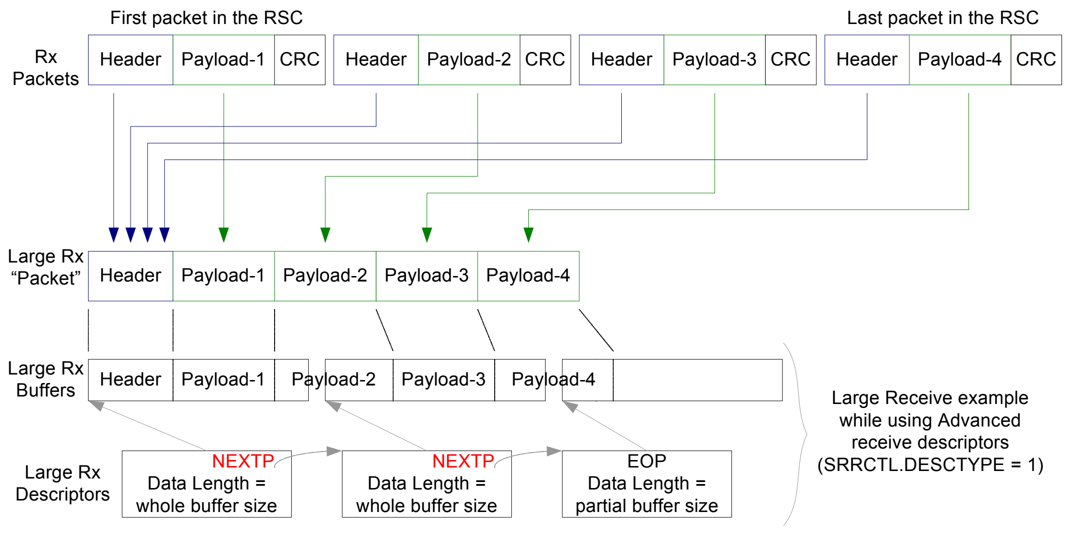

Figure 7-41 RSC Functionality (No Header Split)

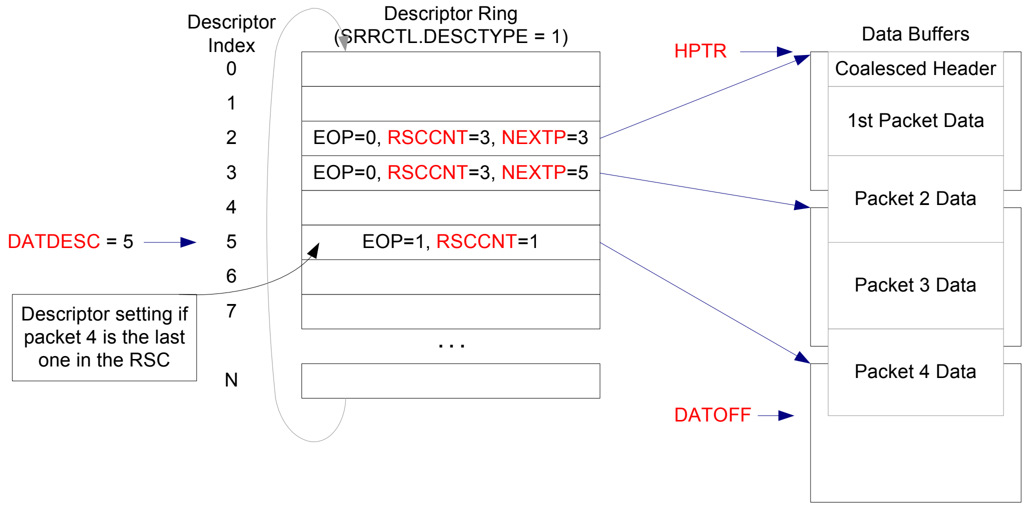

Figure 7-42 RSC Functionality (No Header Split)

Note: Software might abort reception to any queue at any time. For example: VFLR or queue disable. Following these settings, hardware aborts further DMA(s) and descriptor completions. Specifically, active RSC(s) in the specific queue(s) are not completed. In such cases there could be completed packets and RSC(s) hidden from software by prior incomplete RSC(s).

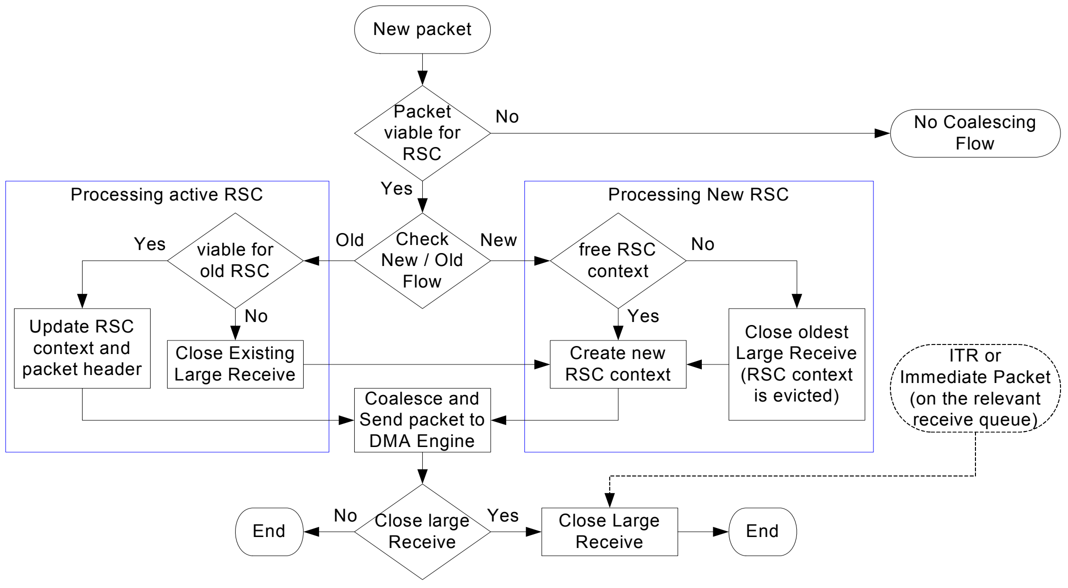

Figure 7-43 RSC Event Flow

7.11.1 Packet Viability for RSC Functionality

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, the received packet is processed in the legacy (non-coalescing) scheme.

- RSC is not disabled globally by the RFCTL.RSC_DIS bit. Note that in SR-IOV mode the RSC must be disabled globally by setting the RFCTL.RSC_DIS bit.
- RSC is enabled in the destination receive queue by the RSCCTL.RSCEN. In this case, software must set the SRRCTL.DESCTYPE field in the relevant queues to advanced descriptor modes.
- The SRRCTL[n].BSIZEHEADER (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- The SRRCTL[n].BSIZEPACKET (packet buffer size) must be 2 KB at minimum.
- The received packet has no MAC errors and no TCP/IP checksum errors. MAC errors are: CRC error or undersize frame received or oversize frame received or error control byte received in mid-packet or illegal code byte received in mid-packet.
- If the *Length* field in the IP header does not cover the entire packet (as the case for padding bytes) then the received packet is not a candidate for RSC.
- If the packet carries LinkSec encapsulation, the LinkSec offload is activated on the packet with no errors.
- The packet type is TCP/IPv4 (non-SNAP) with optional VLAN header.
- IP header does not carry any option headers.
- NFS packets can be coalesced only if NFS filtering is disabled by setting both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits to 1b. Furthermore, the PSR_type1 bit (header split on NFS) must be turned off in all PSRTYPE[n] registers.
- If NFS coalescing is not required, software should set both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits to 0b.
- The packet does not carry IPsec encapsulation (regardless if IPsec offload is enabled).
- The TCP segment is not fragmented.
- The following TCP flags are inactive: FIN, SYN, RST, PSH, URG, ECE, CWR, NS and the other three reserved TCP flags (see TCP Flags mapping in Table 7-76).
- The *ECT* and *CE* bits in the *TOS* field in the IP header are not equal to 11b (see the flags in Table 7-77).
- The packet does not carry any TCP option headers.
- Virtualization rule 1: RSC is not supported for switched packet transmitted from a local VM.
- Virtualization rule 2: When a Rx packet is replicated or mirrored, it might be coalesced only on the Rx queue that belongs to the source VM.
- Note that there are no limitations on the maximum packet length including jumbo packets.

- If there is already an active RSC for the matched flow, then a few additional conditions should be met as listed in Section 7.11.4.

The supported packet format is as follows:

Size	Packet fields
6 Byte	Destination Ethernet MAC address
6 Byte	Source Ethernet MAC address
[8 / 16 Byte]	Optional LinkSec header (supported by RSC only if LinkSec offload is enabled and the hardware extracts this header)
[4 Byte]	Optional VLAN
[4 Byte]	Optional 2nd VLAN (double VLAN)
2 Byte	Ethernet type field equals 0x0800 (MS byte first on the wire)
20 Byte	IPv4 header with no options
20 Byte	Basic TCP header (no options — refer to the rows that follow)
[10 Byte]	Optional TCP time stamp header: 1 Byte Kind 0x08 1 Byte Length 0x0A 4 Byte TS value variable 4 Byte TS echo reply variable
[1 Byte]	Optional TCP no operation header 1 Byte Kind 0x01
[1 Byte]	Optional TCP end of option header list 1 Byte Kind 0x00
Variable length	TCP payload (RSC candidate must have payload size greater than zero)
[8 / 18 Byte]	Optional LinkSec Integrity Checksum Value — ICV (supported by RSC only if LinkSec offload is enabled and the hardware extracts this field)

Table 7-76 Packet Format Supported by RSC

11	10	9	8	7	6	5	4	3	2	1	0
Reserved			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

Table 7-77 IP TOS Field — Bit Map

7	6	5	4	3	2	1	0
TOS (DS)						ECT	CE

Table 7-78 TCP Time-Stamp Option Header (RFC 1323)

1 byte: First on the wire	1 byte	4 byte	4 bytes: Last on the wire
Kind = 0x8	Length = 10	TS Value (TSval)	TS Echo Reply (TSecr)

7.11.2 Flow Identification and RSC Context Matching

TCP/IP packet's flow is identified by its four tuples: Source / Destination IP addresses and Source / Destination TCP port numbers. These tuples are compared against the *Flow Identification* fields stored in the active RSC contexts (listed in Table 7-79). Comparison is done in two phases:

- Hash Compare — Hardware computes a hash value of the four tuples for each flow. The hash value is stored in the RSC context table. It is used for silicon optimization of the compare logic. The hash value of the incoming packet is compared against the hash values of all RSC contexts. No match between the two hash values means that there is no valid context of the same flow.
- Perfect Match — Hardware checks the four tuples of the RSC context that passed the first step with the received frame.
  - A match between the two means that an active RSC context is found.
  - Mismatch between the two indicates a hash collision, which causes a completion of the collided RSC.
- In any case of context mismatch, a new context might be opened as described in Section 7.11.3.
- If the packet's flow matches an active RSC context then the packet might be appended to the existing RSC as described in Section 7.11.4.

Table 7-79 RSC Context

Size	Name	Description
Flow Identification[1]		
1 bit	CVALID	Context valid indication. Set to 1b by hardware when a new context is defined. Cleared to zero when RSC completes.
1 bytes	CHASH	Context hash value (logic XOR of all bytes of the four tuples).
16 bytes	IPDADDR	IP destination address (set to zero for inactive context).
16 bytes	IPSADDR	IP source address (set to zero for inactive context).
1 bit	IP4TYPE	Defines IP version type (set to 1 for IPv4).
2 bytes	TCPDPORT	TCP destination port.
2 bytes	TCPSPORT	TCP source port.