Transmit functionality on the outer VLAN header

- A packet with a single VLAN header is assumed to have only the outer VLAN.
- The outer VLAN header must be added by software as part of the Tx data buffers.
- Hardware does not relate to the outer VLAN header other than the capability of skipping it for parsing inner fields.
- Hardware expects that any transmitted packet (see the disclaimer that follows) has at least the outer VLAN added by software. For any offload that hardware might provide in the transmit data path, hardware assumes that the outer VLAN is present. For those packets that an outer VLAN is not present, any offload that relates to inner fields to the Ethertype might not be provided.

Transmit functionality on the inner VLAN header

- The inner VLAN header can be added by software in one of the following methods:
  - The header is included in the transmit data buffers.
  - The 16-bit portion of the header that includes the priority tag, CFI and VLAN ID are included in the transmit descriptor. The VLAN Ethertype is taken from the *VT* field in the DMATXCTL register.
  - In IOV mode, the priority tag, CFI and VLAN ID can be taken from the PFVMVIR (see details in Section 7.10.3.9.2)
- Hardware identifies and skips the VLAN header for parsing inner fields.
- DCB — The user priority of the packet is taken from the inner VLAN. The traffic class is dictated by the Tx queue.
- Pool Filtering — Destination pool(s) and anti-spoofing functionality is based on the Ethernet MAC address and inner VLAN (if present) as described in Section 7.10.3.4 and Section 7.10.3.9.2.

7.4.5.1 Receive Handling of Packets with VLAN Header(s)

Receive functionality on the outer VLAN header

- If the packet carries a single VLAN header, it is assumed as the outer header and is treated as such.
- Hardware checks the Ethertype of the outer VLAN header against the programmed value in the EXVET register. VLAN header presence is indicated in the Status.VEXT bit in the Rx descriptor. In the case of mismatch, the packet is handled as unknown packet type at which time hardware does not provide any offloads other than LinkSec processing. Also, hardware skips the header for parsing inner fields and provides any supported offload functions.
- The outer VLAN header is posted as is to the receive data buffers.

Receive functionality on the inner VLAN header

- Hardware checks the Ethertype of the inner VLAN header against the programmed value in the VLNCTRL.VET. VLAN header presence is indicated in the Status.VP bit in the Rx descriptor.

- If the RXDCTL.VME is set, the inner VLAN is stripped by hardware while the priority tag, CFI and VLAN ID are indicated in the *VLAN Tag* field in the Rx descriptor.
- Hardware identifies and skips the VLAN header for parsing inner fields and provides any supported offload functions.
- L2 packet filtering is based on the VLAN ID in the inner VLAN header.
- Pool Filtering — Destination pool(s) are defined by the Ethernet MAC address and inner VLAN (if presence) as described in Section 7.10.3.3.
- DCB — The user priority of the packet is taken from the inner VLAN. In the absence of inner VLAN, the packet is assumed as user priority 0 (least priority). See Section 7.4.5.2 for the absence of any VLAN headers.

7.4.5.2 Packets with no VLAN headers in Double VLAN Mode

There are some cases when packets might not carry any VLAN headers, even when extended VLAN is enabled. A few examples for packets that might not carry any VLAN header are: flow control and priority flow control, LACP, LLDP, GMRP, and optional 802.1x packets. When it is expected to transmit untagged packets by software in double VLAN mode the software must not enable VLAN anti-spoofing and VLAN validation nor transmit to receive switching.

Transmitted functionality

DCB — The traffic class in the Tx data path is directed by the Tx queue of the transmitted packet.

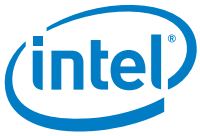
Transmit offload functionality — Software should not enable any offload functions other than LinkSec.

Receive functionality

DCB — Assume user priority 0 (lowest priority).

Receive offload functionality — pool and queue are selected by the Ethernet MAC address or ETQF/ETQS registers. LinkSec offload is functional. Filtering to host and manageability remains functional.

The *Extended VLAN* bit in the CTRL_EXT register and DMATXCTL.GDV are not set. Hardware expects that Rx and Tx packets might not carry a VLAN header or a single VLAN header. Hardware does not relate to the programming of the *VET EXT* field in the EXVET register. Tx and Rx handling of packets with double VLAN headers is unexpected.



7.4.5.3 Packet Priority in Single and Double VLAN Modes

This section summarizes packet handling in both single and double VLAN modes. The user priority of a packet is meaningful in DCB mode when multiple traffic classes are enabled as well as LLIs. The user priority is extracted from the packets as listed in the following table.

Table 7-49 Packet Handling in Single and Double VLAN Modes

Packet type	Single VLAN	Double VLAN
Packet with no VLAN	User priority = 0	User priority = 0
Packet with 1 VLAN	The user priority field in the VLAN header in the packet	User priority = 0
Packet with 2 VLANs	Erroneous case: The user priority field in the **outer** VLAN header in the packet	The user priority field in the **inner** VLAN header in the packet

7.5 Direct Cache Access (DCA)

DCA is a method to improve network I/O performance by placing some posted inbound writes directly within CPU cache. DCA potentially eliminates cache misses due to inbound writes.

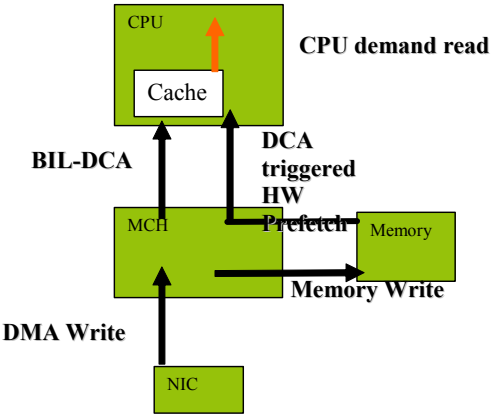


Figure 7-29 Diagram of DCA Implementation on FSB System

As Figure 7-29 illustrates, DCA provides a mechanism where the posted write data from an I/O device, such as an Ethernet NIC, can be placed into CPU cache with a hardware pre-fetch. This mechanism is initialized upon a power good reset. A device driver for the I/O device configures the I/O device for DCA and sets up the appropriate CPU ID and bus ID for the device to send data. The device then encapsulates that information in PCIe TLP headers, in the tag field, to trigger a hardware pre-fetch to the CPU cache.

DCA implementation is controlled by separate registers (DCA_RXCTRL and DCA_TXCTRL) for each transmit and receive queue. In addition, a DCA disable bit can be found in the DCA_CTRL register, and a DCA_ID register can be found for each port, in order to make the function, device, and bus numbers visible to the driver.

The DCA_RXCTRL and DCA_TXCTRL registers can be written by software on the fly and can be changed at any time. When software changes the register contents, hardware applies changes only after all the previous packets in progress for DCA have completed.

However, in order to implement DCA, the 82599 has to be aware of the IOAT version used. The software driver initializes the 82599 to be aware of the bus configuration. The *DCA Mode* field in the DCA_CTRL register defines the system configuration:

1. Legacy DCA: The DCA target ID is derived from CPU ID.
2. DCA 1.0: The DCA target ID is derived from APIC ID.

Both modes are described as follows.

7.5.1 PCIe TLP Format for DCA

Figure 7-30 shows the format of the PCIe TLP for DCA.

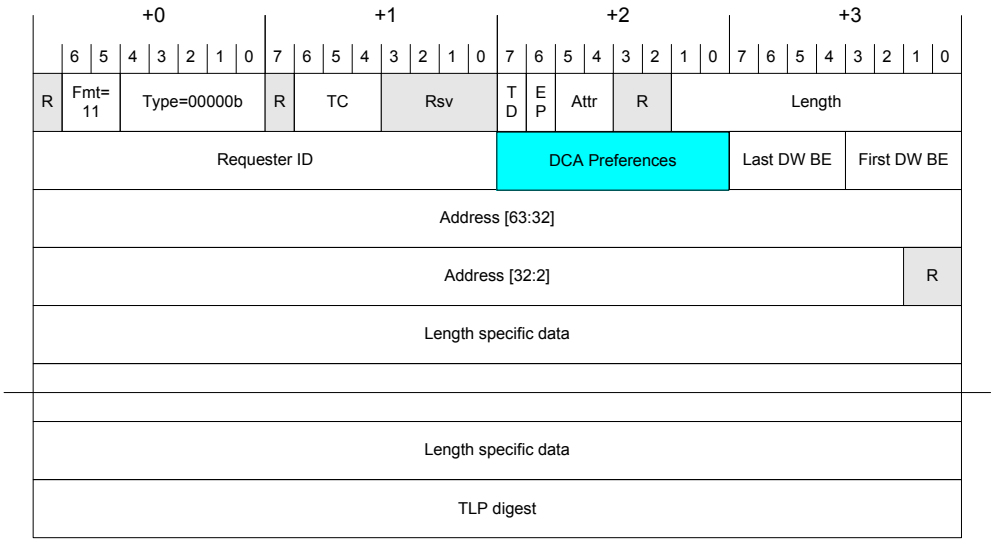


Figure 7-30 PCIe Message Format for DCA

The DCA preferences field has the following formats.

For legacy DCA systems:

Bits	Name	Description
0	DCA indication	0b = DCA disabled. 1b = DCA enabled.
4:1	DCA target ID	The DCA target ID specifies the target cache for the data.

For DCA 1.0 systems:

Bits	Name	Description
7:0	DCA target ID	0000.0000b: DCA is disabled. Other: Target core ID derived from APIC ID.25915

Note: All functions within a the 82599 have to adhere to the tag encoding rules for DCA writes. Even if a given function is not capable of DCA, but other functions are capable of DCA, memory writes from the non-DCA function must set the tag field to 00000000b.

7.6 LEDs

The 82599 implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indication.

The configuration for LED outputs is specified via the LEDCTL Register. Furthermore, the hardware-default configuration for all LED outputs can be specified via EEPROM fields thereby supporting LED displays configurable to a particular OEM preference.

Each of the four LED's can be configured to use one of a variety of sources for output indication. For more information on the MODE bits see LEDCTL register (see Section 8.2.3.1.6).

The *IVRT* bits enable the LED source to be inverted before being output or observed by the blink-control logic. LED outputs are assumed to normally be connected to the negative side (cathode) of an external LED.

The *BLINK* bits control whether the LED should be blinked (on for 200 ms, then off for 200 ms) while the LED source is asserted. The blink control can be especially useful for ensuring that certain events, such as ACTIVITY indication, cause LED transitions, which are sufficiently visible by a human eye.

Note: The LINK/ACTIVITY source functions slightly different from the others when BLINK is enabled. The LED is:

- Off if there is no LINK
- On if there is LINK and no ACTIVITY
- Blinks if there is LINK and ACTIVITY

7.7 Data Center Bridging (DCB)

See Section 4.6.11 for the DCB configuration sequence and Section 11.5 for the expected performance of DCB functionality.

7.7.1 Overview

DCB is a set of features that improve the capability of Ethernet to handle multiple traffic types (such as LAN, storage, IPC) by answering the various needs of those types. DCB enables multiple traffic types that have different requirements of packet delivery, bandwidth allocation and delay. Each traffic type can have one or several user priorities, or Traffic Classes (TCs). For example, IPC might have a high priority class for synchronization messages between servers and lower priority traffic class for bulk traffic exchange between servers. Most of the DCB functions impact the transmit traffic generated from the end node to the network (traffic generation). The receive data path needs be compliant with the requirements of DCB and provide the required functions as a traffic termination point.

DCB system requirements include:

1. Bandwidth grouping — For effective multiplexing that simulates a separate link for the separate types of traffic, DCB requires that traffic types be recognized as groups in the bandwidth and priority handling by nodes in the network. Traffic types are associated to Bandwidth Groups (BWGs). The system needs to be able to allocate bandwidth to the BWGs in a way that emulates that group being on its own separate link.
2. Bandwidth fairness — DCB multiplexing functions (transmit) and de-multiplexing functions (receive) need to guarantee minimum allocation of bandwidth to traffic types and traffic classes. Fairness between groups comes first, then fairness between TCs. If system resources (such as PCIe bandwidth) limit total throughput, then the available bandwidth should be distributed among consumers proportionally to their allocations.
3. Latency of operation — DCB multiplexing and de-multiplexing functions need to allow minimum latency for some TCs. Arbitration mechanisms, packet buffers, descriptor queues and flow control algorithm need to be defined and designed to allow this. The best example is the control/sync traffic in IPC. The expectation for end-to-end IPC control is measured in the low 10's of µs for the 82599 and is expected to drop to a singe digit µs later. Some elements in multimedia traffic also bear similar requirements. Although some of the end-to-end delays can be quite long, the individual contribution of the arbitration in each node must be kept to a minimal. End-to-end budgets do not comprehend large delays within transmission nodes.
4. No-drop behavior and network congestion management — The end node must be able to guarantee no-drop behavior for some TCs or some packets within TCs. As a termination point in receive, it is the end node’s responsibility to properly control traffic coming from the network to achieve this end. For traffic generation in transmit, the end station must be able to positively respond to flow control from the network as the must have tool to prevent packet drop. It also needs to participate in network congestion management.

5. Compatibility with existing systems. — The DCB implementation needs to be usable by IT using known configurations and parameters, unless new ones are made expressly available. For example, DCB implementation cannot assume new knowledge regarding bandwidth allocation of traffic types that do not have known bandwidth requirements.

The layer 2 features of DCB implemented in the 82599 are:

1. Multi-class priority arbitration and scheduling — The 82599 implements an arbitration mechanism on its transmit data path. The arbitration mechanism allocates bandwidth between TC in BWGs and between Virtual Machines (VMs) or Virtual Functions (VFs) in a virtualization environment. The BWGs can be used to control bandwidth and priority allocated to traffic types. Typically BWGs should be used to represent traffic types. TC arbitration allows control of bandwidth and priority control within BWGs as well as within the entire link bandwidth. The arbitration is designed to respect the bandwidth allocations to BWGs. The priority allocation allows minimization of delay for specific TCs. In the 82599, TCs and user priorities are processed on a packet-by-packet basis based on the 802.1p identifier in the 802.1Q-tag.
2. Class-based flow control (PFC — Priority Flow Control) — Class-based flow control functionality is similar to the IEEE802.3X link flow control. It is applied separately to the different TCs.
   - Transmit response to class-based flow control from the ingress switch it is connected to.
   - Receive class-based flow control commands to the switch in response to packet buffers filling status.
3. DMA queuing per traffic type — Implementation of the DCB transmit, minimization of software processing and delays require implementation of separate DMA queues for the different traffic types. The 82599 implements 128 descriptor queue in transmit and 128 descriptor queues in receive.
4. Multiple Buffers — The 82599 implements separate transmit and receive packet buffers per TC.
5. Rate-limiter per Tx queue — limiting the transmit data rate for each Tx queue.

Latency requirements:

Quantitative latency requirements are defined for a single 64-byte packet at the highest priority traffic class. Latency is defined separately for transmit and receive:

- Transmit latency — measured from a tail update until the packet is transmitted on the wire. It is assumed that a single packet is submitted for this TC and its latency is then measured in the presence of traffic belonging to other TCs.
- Receive latency — measured from packet reception from the wire and until the descriptor is updated on PCIe.

Figure 7-31 shows the latency requirements as previously defined.

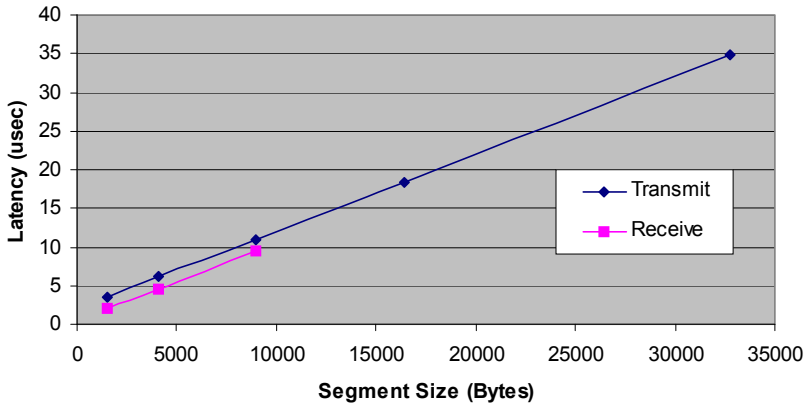


Figure 7-31 Latency Requirements

Note: In DCB mode, it is assumed all traffic is tagged (contains a VLAN header) — except for Layer2 frames with special Ethernet MAC addresses that goes untagged. GMRP frames (special Ethernet MAC addresses starting with 0x0180c20000) must, however, go tagged. Untagged packets must be delivered to the host and are assumed to belong to User Priority 7.

Note that any BCN signaling is terminated at the network's edge. At initialization, every component exchanges its capabilities with its peer via a Capability Exchange (DCX) protocol carried over dedicated Link Layer Discovery Protocol (LLDP) frames. Support for these protocols is transparent to the hardware implementation, and as a result, is not described in this document.

7.7.2 Transmit-side Capabilities

Note: When configured for DCB mode, the the 82599 driver should only use advanced transmit descriptors. Refer to Section 7.2.3.2.3. Using legacy transmit descriptors is not allowed.

7.7.2.1 Transmit Rate Scheduler (RS)

7.7.2.1.1 Basic Rate Control Operation

Rate control is defined in terms of maximum payload rate, and not in terms of maximum packet rate. This means that each time a rate controlled packet is sent, the next time a new packet can be sent out of the same rate controlled queue is relative to the packet size of the last packet sent. The minimum spacing in time between two starts of packets sent from the same rate controlled queue is recalculated in hardware on every packet again, by using the following formula:

```
MIFS = PL x RF
```

Where:

- Packet Length (PL), is the Layer2 length (such as without preamble and IPG) in bytes of the previous packet sent out of that rate controller. It is an integer ranging from 64 to 9 K (at least 14-bits).
- RF = 10 Gb/s / target rate (rate factor) is the ratio between the nominal link rate and the target maximum rate to achieve for that rate-controlled queue. It is dynamically updated either by software via the RTTBCNRC register, or by hardware via the rate-drift mechanism, as described in Section 7.7.2.1.2. It is a decimal number ranging from 1 to 1,000 (10 Mb/s minimum target rate). For example, at least 10-bits before the hexadecimal point and 14-bits after as required for the maximum packet length by which it is multiplied. For links at 1 Gb/s, the rate factor must be configured relatively to the link speed, replacing 10 Gb/s by 1 Gb/s in the above formula.
- Minimum Inter Frame Space (MIFS) is the minimum delay in bytes units, between the starting of two Ethernet frames issued from the same rate-controlled queue. It is an integer ranging from 76 to 9,216,012 (at least 24-bits). In spite of the 8-byte resolution provided at the internal data path, the byte-level resolution is required here to maintain an acceptable rate resolution (at 1% level) for the small packets case and high rates.

Note: It might be that a pipeline implementation causes the MIFS calculated on a transmitted packet to be enforced only on the subsequent transmitted packet.

Time Stamps — A rate-scheduling table contains the accumulated interval MIFS, for each rate-controlled descriptor queue separately, and is stored as an absolute Time Stamp (TS) relative to an internal free running timer. The TS value points to the time in the future at which a next data read request can be sent for that queue. Whenever updating a TimeStamp:

```
TimeStamp(new) = TimeStamp(old) + MIFS
```

When a descriptor queue starts to be rate controlled, the first interval MIFS value is equal to 0 (TS equal to the current timer value) — without taking into account the last packet sent prior to rate control. When the TS value stored becomes equal to or smaller than the current free running timer value, it means that the switch is on and that the queue starts accumulating compensation times from the past (referred as a negative TS). When the TS value stored is strictly greater than the current free running timer value, it means that the switch is off (referred as a positive TS.

```
(CurrentTime) < TimeStamp       <-->    switch is “off”

(CurrentTime) >= TimeStamp      <-->    switch is “on”
```

MMW — The ability to accumulate negative compensation times that saturates to a Max Memory Window (MMW) time backward. MMW size is configured per TC via the *MMW_SIZE* field of the RTTBCNRM register, and is expressed in 1 KB units of payload, ranging from 0 up to 2 KB units (at least 11-bits). The MMW_SIZE configured in KB units of payload has to be converted in time interval MMW_TIME expressed in KB, before a new time stamp is checked for saturation. It is computed for each queue according to its associated Rate Factor (RF) using the following formula:

```
MMW_TIME = MMW_SIZE x RF
```

Note: MMW_TIME is rounded by default to a 1 KB precision level and must be at least 31 bits long. Hence, the time stamp byte-level values stored must be at least 32-bits long for properly handling the wrap-around case. 29-bits are required for the internal free running timer clocked once every 8 bytes.

Whenever updating a time stamp verify:

```
TimeStamp(old) + MIFS >= (CurrentTime) – MMW_TIME
```

and then the time stamp is updated according to the non-saturated formula:

```
TimeStamp(new) = TimeStamp(old) + MIFS
```

Otherwise, enforced saturation by assigning:

```
TimeStamp(new) = (CurrentTime) – MMW_TIME + MIFS
```

Note: Non-null MMW introduces some flexibility in the way controlled rates are enforced. It is required to avoid overall throughput losses and unfairness caused by rate-controlled packets over-delayed, consequently to packets inserted in between. Between two rate-limited packets spaced by at least the MIFS interval, non-rate-limited packets, or rate-limited packets from other rate-controlled queues, might be inserted. If a rate controlled packet has been delayed by more time than it was required for rate control (because of arbitration between VMs or TCs), the next MIFS accumulates from the last time the queue was switched on by the rate scheduling table — and not from the current time. Refer to Figure 7-32 for visualizing the effect of MMW.

MMW_SIZE set to 0 must be supported as well.

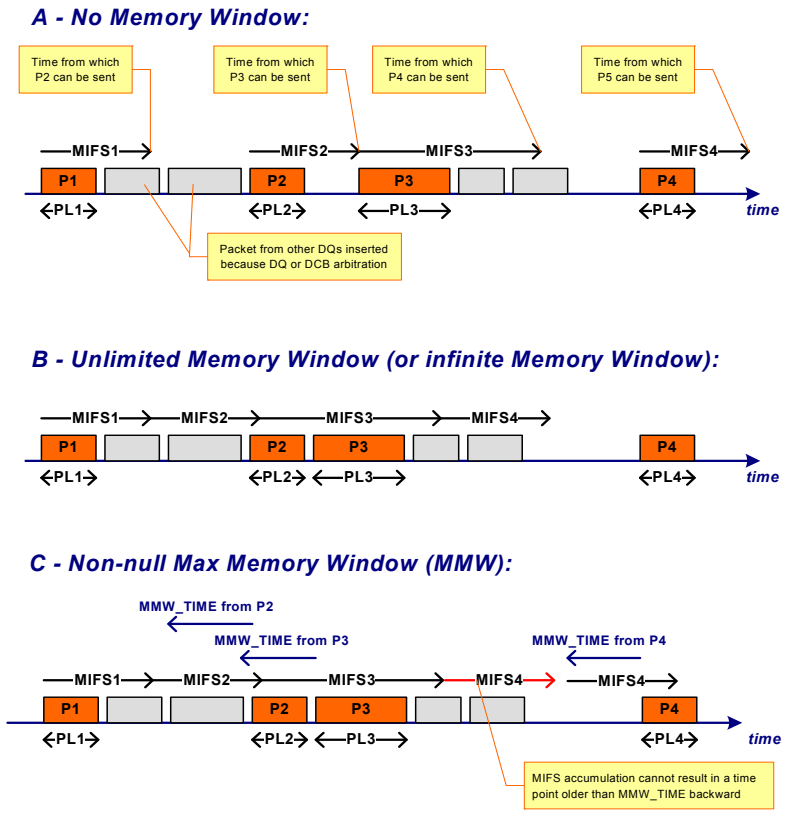


Figure 7-32 Minimum Inter-Frame Spacing for Rate-Controlled Frames (in Orange)

7.7.2.1.2 Rate Drift

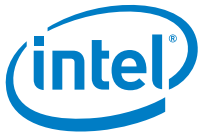
Periodically, at fixed intervals in time, the rate factors of all rate-controlled queues must be increased internally by a small amount. The periodic interval in time at which rate drift mechanism is triggered is configured via the *DRIFT_INT* field in RTTBCNRD register. the rate-drift mechanism done in hardware is enabled by setting the *DRIFT_ENA* bit in RTTBCNRD register; otherwise, it is assumed that it is handled by software.

The rate-drift mechanism is essential for fairness and rate recovery of rate-controlled flows reduced to very low rates.

Note: For providing accurate rate-drift intervals, the rate-drift mechanism must be started immediately once the interval in time has elapsed — without waiting for the next time stamp table scan cycle to start.

Rates are increased in a multiplicative manner, by multiplying the rates with a fixed value slightly *greater* than unity. It is thus similar to multiplying the rate factors by a fixed value slightly *smaller* than unity, which is referred to as the drift factor. It is configured via the *DRIFT_FAC* field in RTTBCNRD register. The rate-drift mechanism saturates if the full line rate has been recovered (when the rate factor has been decreased down to unity):

```
Rate-Factor(new) = max(1, Rate-Factor(old) x Drift-Factor)
```



For example, if a periodic rate increase of 3% is desired, then a drift factor of 1/ 1.03=0.97087... must be configured.

Note: One disadvantage of the multiplicative rate increase method results in smaller increases for low-rated flows and larger increases for high-rated flows. An additive method has been envisaged instead, increasing the rate factors by small additive steps on each interval, but it has been dropped off because it had poor chances of being standardized by IEEE 802.1au.

A queue that has recovered the full line rate via the rate-drift mechanism (rate factor decreased down to one) is not considered as a rate-controlled queue, its corresponding *RS_ENA* bit in the RTTBCNRC register must be internally self-cleared, and it should stop tagging its frames with the RLT option. Refer to Section 7.7.2.1 for further details on CM-tagging.

7.7.2.2 User Priority to Traffic Class Mapping

DCB-enabled software is responsible for classifying any Tx packet into one of the eight 802.1p user priorities, and to assure it is tagged accordingly by either software or hardware. The driver dispatches classified Tx traffic into the Tx queues attached to the proper TC, according to a UP-to-TC Tx mapping policy decided by the IT manager.

Note: When configured for DCB mode or when using the Tx rate-limiting functionality, the 82599 software driver should only use advanced transmit descriptors. Refer to Section 7.2.3.2.3. DO NOT use legacy transmit descriptors.

Caution: When translating XON/XOFF priority flow control commands defined per UP into commands to the Tx packet buffers, the 82599 is required to use the same UP-to-TC Tx mapping table that software is using. The RTTUP2TC register must be configured by software accordingly. Refer to Section 3.7.7.1.3 for details on priority flow control.

7.7.2.3 VM-Weighted Round-Robin Arbiters

The 82599 implements VM-weighted arbiter(s) for virtualized environments and according to the following case:

- If DCB is enabled, there is one such arbiter per TC, arbitrating between the descriptor queues attached to the TC (one queue per VF). Bandwidth allocation to VMs is enforced at the descriptor plane, per each TC separately. The VM arbiter instantiated for each TC is aimed to elect the next queue for which a data read request is sent in case the TC is elected for transmission by the next level arbiter. For example, the TC weighted strict priority descriptor plane arbiter.
- If DCB is disabled, there is one single VM weighted arbiter, arbitrating between pools of descriptor queues, where a pool is formed by the queues attached to the same VF. Bandwidth allocation to VMs is enforced at the descriptor plane, between the pools, where queues within a pool are served on a frame-by-frame round-robin manner.

Refer to the different arbitration schemes where virtualization is enabled, as shown in Figure 7-17.

Note: In this section, VM is considered a generic term used to refer to the arbitrated entity, whether it is a Tx descriptor queue within the TC or whether it is a pool of Tx descriptor queues. In the later case, pool parameters are allocated only to the lowest indexed queue within the pool, taken as the representation of the entire pool.

7.7.2.3.1 Definition and Description of Parameters

Credits: Credits regulate the bandwidth allocated to VMs. As part of the Weighted Round Robin (WRR) algorithm, each VM has pre-allocated credits. They are decremented upon each transmission and replenished cyclically. The ratio between the credits of the VMs represents the relative bandwidth percentage allocated to each VM (within the TC for the DCB enabled case). The 82599 effectively maintains one table that represents these ratios. Note that credits can get negative values down to the maximum sized frame allowed on the TC/pool.

Note: The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, since credits can accumulate only up to twice the credit refills, the refills should be allocated as low as possible but must be set greater than the maximum sized frame allowed on the TC or on the pool.

WRR: The algorithm implemented in the 82599 for VM arbitration.

Table 7-50 (T1) defines the VMs and their bandwidth allocation. The following elements are defined in this table:

Table 7-50 Bandwidth Allocation to VMs

T1: VM Bandwidth Allocation			
VM_N	VM Refill	VM Max Credits	VM Min Credits
0	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
1	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
2	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
3	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
...			
15	MSS:1,024 KB	2xMSS:2,048 KB	-MSS
† Due to a pipelined implementation, the VM credits range is enlarged by one MSS, beyond negative limits.			
†† All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).			

VM: Configuration — The unique Tx descriptor queue attached to a VF within a TC, or the pool of Tx descriptor queues attached to the same VF.

VM Credit Refill: Configuration — The 82599's WRR algorithm implement credit refill as the technique for percentage allocation to VMs. The credits refill are added to each VM credit count on each completion of the full round of the algorithm (after all the VMs had their chance to send what they had in store or at least one frame).

The 82599's driver needs to calculate the VM credit refill to match the percent allocated through management (such as in the MIB). Since the WRR arbitration is self timed, the ratio between the credits refill is the only defining parameter for the VMs. However, the refills must be greater or equal to the maximum sized frame allowed on the TC or on the pool in order to guaranty transmission of at least one frame on each recycling round. The 82599 allows a value of 1.5 KB to 1,024 KB for a dynamic range of x1000.

VM Maximum Accumulated Credits: Deducted from Refill Configuration — In order to prevent the use of stale credits, the number of credits each VM can accumulate upon refill is limited. The credits for each VM can only reach twice their refill. The maximum range for the credits is thus -9.5 KB to 2,048 KB, assuming negative credits can accumulate up to a maximum sized frame (9.5 KB [9728 bytes] if jumbo frames are allowed), and where positive credits can accumulate up to twice the maximum credit refill.

VM Credits: Run time parameter — VM credits is a running counter for each VM. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the VMs and enables transmission for those VMs that have enough pending credits (their credit number is greater than zero).

Table 7-51 Registers Allocation for Tx VM Arbiters

Attribute	Tx VM Arbiter
VM Control registers	RTTDT1C
VMC Status registers	RTTDT1S
VM credit refill	CRQ
VM credits	CCC

7.7.2.3.2 WRR Arbiter Algorithm

Round Robin — The round-robin aspect of the VM WRR arbiter resides in the fact that once a VM has been granted for a data read request, the next VMs are checked in a cyclic round-robin order, even if the granted VM still has credits for another data read request.

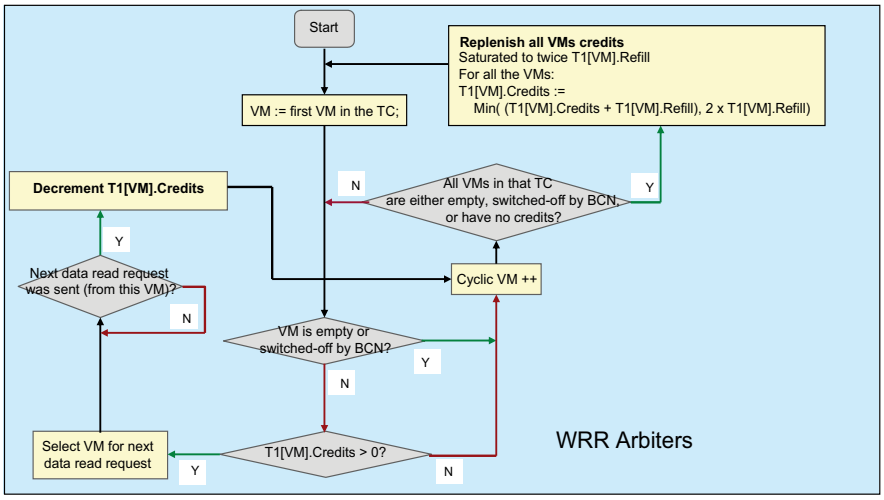


Figure 7-33 Tx VM WRR Arbiters Operation

7.7.2.4 Tx TC Weighted Strict Priority Arbiters

In DCB, multiple traffic types are essentially multiplexed over the Ethernet 10 Gb/s network. There is a need to allow different behavior to different traffic flows as they pass through multiple Ethernet switches and links. For example, for LAN, SAN and IPC connections are consolidated on a single Ethernet link. Each traffic type (BWG) is guaranteed the bandwidth it has been allocated and is prevented from usurping bandwidth from other types. However, if a BWG does not use its bandwidth, that bandwidth is made available to the other BWGs. The same holds for TCs within a BWG. If allocated some bandwidth, TCs are guaranteed to have it, and if unused, that bandwidth can be used by the other TCs within the BWG. Information regarding bandwidth allocation for some TCs might not be available. In the case of LAN, the entire allocation of bandwidth within the LAN link is typically undefined in today's networks. The arbitration scheme includes Group Strict Priorities (GSP) to cover for that. TCs for which the *GSP* bit is set are limited by the total throughput allocated to their BWG rather than to TC allocation.

Link bandwidth is divided among the BWGs for guaranteed minimum behavior. For example: LAN:
4 Gb/s, SAN: 4 Gb/s, IPC: 2 Gb/s. The 82599 supports two types of bandwidth allocation within BWGs. TCs can be either allocated bandwidth or be used as in strict priority. If a TC does not use all of its allocated bandwidth, that bandwidth is recycled to other TCs in the BWG.

The 82599 implements two replications of the weighted TC arbiter:

- One in the descriptor plane, arbitrating between the different descriptor queues, deciding which queue is serviced next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is smaller than the link bandwidth.
- A second in the packet plane, at the output of the packet buffers, deciding which packet to transmit next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is greater than the link bandwidth.

The condition for entry into the bandwidth allocation algorithm sequence differs for the descriptor and data arbiters:

- The descriptor arbiter queries whether there is at least one queue attached to a TC that is not empty, not switched off by the rate scheduler, with positive VM weighted arbiter credits (when relevant), and for which the destined packet buffer has room for the worst case maximum sized frame. This last condition is controlled by RTTDCS.BPBFSM.
- The packet arbiter queries whether the packet buffer has a packet to send and whether it is not stalled by priority flow control.

7.7.2.4.1 Definition and Description of Parameters

User Priority (UP): There are eight traffic priorities, determined by 802.1p tag bits on an Ethernet link. The Q-Tag field holds UP's. Per 802.1p, Priority #7 is the highest priority. User priorities are assigned by the application or the system to certain usage classes, such as manageability, IPC control channel, VoIP. An additional bit within the VLAN TCI field (bit 5 - DEI)defines whether the packet has a no drop requirement. This bit is not being used by DCB mechanisms.

User Bandwidth Group (UBWG): a user bandwidth group is a management parameter that is a binding of user priorities into bandwidth groups for provisioning purposes. The hardware implementation does not recognize the UBWG entity.

Traffic Class (TC): incoming packets are placed in traffic classes. Per the DCB functional specification, there might be a 1:1 mapping between UP and TC, or more than one priority can be grouped into a single class. Such grouping does not cross boundaries of traffic BWGs. the 82599 implements eight or four TCs and maps them to UP's according to a programmable register. This provides the best flexibility for the IT manager. However, when more than one UP is mapped to the same TC, they must have the same no-drop policy network wide.

Packet Buffer (PB): TCs are mapped to packet buffers in a straightforward 1:1 mapping. Packets are also placed in packet buffers based on their class assignments.

Traffic Bandwidth Group (BWG): For bandwidth allocation and grouping, one or more TC can be grouped into a Traffic Bandwidth Group (BWG). A BWG is a logical association at a node, and has no markings inside a packet header. End stations and switches are independent in their definition and allocation of grouping of different TCs. Consistency of behavior throughout the network is handled by the UBWG provisioning mechanism.

One or more TCs can be grouped in a BWG. BWGs are allocated a percentage of bandwidth of available Ethernet link. The allocated bandwidth for BWG can be further divided among the TCs that are inside the BWG.

Credits: Credits regulate the bandwidth allocated to BWGs and TCs. As part of the WSP algorithm, each BWG and TC has pre-allocated credits. Those are decremented upon each transmission and replenished cyclically. The ratio between the credits of the BWGs represents the relative bandwidth percentage allocated to each BWG. The ratio between the credits of the TCs represents the relative bandwidth percentage allocated to each TC within a BWG. The 82599 effectively maintains one table that represents both ratios at once. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Maximum Credit Value: The maximum credit value establishes a limit for the running sum of credits allotted to a class or group. This value prevents stacking up stale credits that can be added up over a relatively long period of time and then used by TCs all at once, altering fairness and latency.

Note: The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, the absolute value might have substantial impact on the algorithm behavior. Larger absolute values can impact the latency of high priority queues and their ability to serve bursts with minimum latency, whereas too small credit values might impact the correct functionality in presence of jumbo frames. The speed of the algorithm implementation should also be taken into account. The value of the maximum credit limit are also in principle not part of the main WSP algorithm. However, they impact the fairness of bandwidth reallocation between queues in case some queues do not transmit the full amount they have been permitted to. Also, small values prevent correct functionality of jumbo frames. From high-level simulations it appears that credits should be allocated as low as possible based on the speed of the algorithm. Maximum credit values for TCs should be 1.5x to 2x the size of the maximum entity expected in that TC.

Weighted Strict Priority (WSP): The algorithm implemented in the 82599 for TC arbitration.

Group Strict Priority (GSP): Refer to the sections that follow for details.

Link Strict Priority (LSP): Refer to the sections that follow for details.

Table 7-52 (T2) defines the TCs, their association to BWGs and their bandwidth allocation. The following elements are defined in this table:

Table 7-52 Bandwidth Allocation to TCs and BWGs

T2: Traffic Class Bandwidth Allocation Within a BWG						
TC_N	BWG	TC Refill	TC Max Credits	LSP	GSP	TC Min Credits (according to GSP 0/1)
0	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
1	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
2	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
3	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
4	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
5	3	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
6	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	1/1	-MSS / -2,048 KB
7	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB

† Due to a pipelined implementation, the TC credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

†† All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

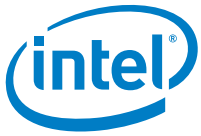
TC: Configuration — The traffic type associated to the packet buffer where incoming packets are kept before transmission (or discard — not implemented in transmit in the 82599). TC7 is the highest priority TC.

BWG: Configuration — Traffic BWG that a TC belongs to.

TC Credit Refill: Configuration — The 82599's WSP algorithm implements credit refill as the technique for traffic class percentage allocation. The credits refill are added to each TC credit count on each completion of the full round of the algorithm (after all TCs had their chance to send what they had in store and if they had credits for it).

The 82599's driver needs to calculate the TC Credit refill to match the percentage allocated through management (in the MIB). The TC credit refill table includes, in one table both the TC and the BWG. Since the WSP arbitration is self timed, the ratio between the credits refill is the only defining parameter for the TC and BWG. The absolute values of the refill have significance as to the rate of distribution between the queues and can have impact on latency and on the momentary stability of the bandwidth fairness. Results from simulations indicate that the quantum for the refill should be small to prevent large swings. It should not be too small as to create overhead in the mechanism due to the execution time; however. The 82599 allows a value of 64 bytes to 32 KB, A dynamic range of x500. The usage model is likely to call for a smallest refill value of 256 bytes to 512 bytes. This leaves a dynamic range of 80x-200x. For example, if a queue is assigned 1% and this is translated into a 256-byte increment, another assigned with 99% would have a refill value of 25344 bytes.

TC MaxCredit: Configuration — In order to prevent use of stale credits, the number of credits each TC can accumulate upon refill is limited. The TC credit can only reach MaxCredit, beyond that its value gets recycled to other queues. Refer to the recycling mode for more details. The maximum range for the MaxCredit is 256 KB. This high range value was inherited from the 82598 that had to deal with entire LSOs, but is not really necessary for the 82599.



Note: Full testing of many values for maximum value is unnecessary. Hierarchical testing should be applied. For the random test, four to six values should be sufficient. It is important that the testing includes values that are relevant to the interesting zone of maximum value. For example, in the 0.8x to 2x the relevant largest entity in the class. Although class with an expected shorter packet could use a smaller MaxCredit, it is recommended that testing fully covers the cases where all the classes have similar values of MaxCredit, as it is a possible variant use of the algorithm.

Link Strict Priority (LSP): Configuration — If set, this bit specifies that this TC can transmit without any restriction of credits. This effectively means that this TC can take up the entire link bandwidth, unless preempted by higher priority traffic. If this bit is set, then TC.CreditRefill must be set to 0b to ensure fair bandwidth allocation. Preferably, the algorithm implementation should disregard non-zero values in all its calculations.

Group Strict Priority (GSP): Configuration — This bit defines whether strict priority is enabled or disabled for this TC within its BWG. If TC.GSP is set to 1b, the TC is scheduled for transmission using strict priority. It does not check for availability of TC.Credits. It does check whether the BWG of this TC has credits (such as the amount of traffic generated from this TC is still limited by the BWG allocated for the BWG (T3. BWGP). If this bit is set, then TC.CreditRefill values can be set to 0b, if a non-zero value is configured, TC credits are reduced first from the GSP TC and if reached to zero from other TCs in the group, if the refill credits are configured to zero the TC credits are reduced from the other TCs in the BWG.

Note: Since the TC.GSP parameter relates to individual TCs, some BWGs might have both TC's with bandwidth allocation and TC's with GSP. This is a hybrid usage mode that is complex to validate and is possibly secondary in importance.

Usage Note — It is possible that a TC using LSP dominates the link bandwidth if there are no packets waiting and eligible in higher priority TC's. To guarantee correct bandwidth allocation, all TC's with the unlimited bit set should be in the same traffic BWG (high priority group). Note that this is different than a typical DCB deployment considered where BWG is created with functional grouping, like LAN, SAN, and IPC etc. TC's with the *LSP* bit set should be the first to be considered by the scheduler (the first TC's). For example, from queue 7 to queue 5 with the other five TC's for groups with bandwidth allocation. These are not strict requirements, if these rules are not followed, undesirable behavior could occur in some cases. A group containing only TC's with the unlimited bit set effectively has zero BWG credits since TC's with the LSP unlimited bit set should have TC credits set to zero. Use of LSP / TC.GSP should be restricted to UP/TC's that service trusted applications.

TC Credits: Run-time parameter — TC credits is a running algebraic counter for each TC. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the TC's and enables transmission for those TCs that have positive pending credits. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Table 7-53 (T3) defines the hierarchy of BWG similarly to T2 defining the hierarchy within the BWG's. T3 implementation should include eight rows.

Table 7-53 Line Bandwidth Allocations to Bandwidth Groups (BWG) (with example)

T3: Line bandwidth allocation to Traffic Bandwidth Groups (BWG)				
BWG	**BWG Refill Credits**	**BWG MaxCredit**	**BWG Credit**	**Description**
0	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	IPC
1	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	SAN
2	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	LAN
3	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048 KB	Manageability
-				
-				
-				
7				

† Due to a pipelined implementation, the BWG credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

BWG: Configuration — This is the number of the traffic BWG that is three bits wide. This field corresponds to the TC.BWG field in Table 7-52.

BWG Refill Credits: A virtual number — The credits provisioned for this BWG. The credits ratio between the BWG's should reflect the ratio of bandwidth between the BWG's. In the actual implementation, this number is the sum of the credit Refills of the TC's associated with this BWG.

BWG MaxCredit: A virtual number — The maximum credits for a BWG. Credits in the BWG.Credit counter are limited to this value. Credits that should have been refilled above this value are lost. In effect, due to the self-timed cyclic nature of the WSP algorithm, those credits are distributed between all BWG's. In the actual implementation, this number is the sum of the MaxCredit of the TC's associated with this BWG.

BWG.Credit: Run-time parameter — A running algebraic counter that is decremented for each transmission. At the end of each cycle, this counter is synchronized with the sum of the TC.Credit counter associated with this BWG. The synchronization algorithm depends on the recycling mode. refer to the sections that follow for details about arbitration configurations. Note that credits can get negative values down to the maximum sized frame allowed on the BWG.

7.7.2.4.2 Arbiters Conventions

The WSP scheme previously described is written with the data plane arbiter in mind. However, the same scheme is used by the transmit descriptor plane arbiter and a subset of it is used by the receive data arbiter. To distinguish between the two arbiters, attributes of the each arbiter are prefixed as depicted in Table 7-54.

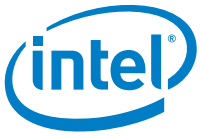


Table 7-54 Attributes of Tx Arbiters

Attribute	Tx Packet Arbiter	Tx Descriptor Arbiter
TC	P-TC	D-TC
BWG	P-BWG	D-BWG
TC Credit Refill	P-TC Credit Refill	D-TC Credit Refill
TC MaxCredit	P-TC MaxCredit	D-TC MaxCredit
LSP	P-LSP	D-LSP
GSP	P-GSP	D-GSP
TC Credits	P-TC Credits	D-TC Credits
BWG Refill Credits	P-BWG Refill Credits	D-BWG Refill Credits
BWG MaxCredits	P-BWG MaxCredits	D-BWG MaxCredits
BWG Credits	P-BWG Credits	D-BWG Credits

Table 7-55 lists the register fields that contain the relevant attributes from Table 7-54.

Table 7-55 Registers Allocation for Tx TC Arbiters

Attribute	Tx Packet Arbiter	Tx Descriptor Arbiter
TC Control registers	RTTPT2C	Reserved
TC Status registers	RTTPT2S	RTTDT2S
BWG	BWG	BWG
TC credit refill	CRQ	CRQ
TC MaxCredit	MCL	MCL
LSP	LSP	LSP
GSP	GSP	GSP
TC credits	CCC	CCC

7.7.2.4.3 Tx TC WSP Arbitration Configurations

RR / WSP: Global Configuration bits — When this bit is set, the arbitration is in WSP mode. When reset, the arbitration is in flat frame-based round robin mode. In RR mode, one frame is transmitted from each packet buffer in its turn. BWG and TC parameters do not apply.

Recycle Mode: Global Configuration bits.

Architecture Overview of Recycle — As a result of GSP transmits and TCs that reach their maximum credit limit, the credit count of a BWG might not match the total credit count of its TCs (refer to the sections that follow for more details). It is not merely an arithmetic issue. The WSP algorithm, dual hierarchy behaves as a maximum allocation algorithm within the BWG's and minimum allocation algorithm between the BWG's. Since the recycle is self timed, when a BWG does not transmit all of its allocated bandwidth within a cycle, at the end of the cycle its bandwidth is in effect reallocated to all BWG's. This results in a minimum allocation behavior. Inside the BWG's; however, this notion of self timing does not exist. Some explicit mechanism is required to recycle bandwidth within a BWG rather than to all the BWG's — The requirement to have a minimum allocation behavior.

- A BWG credit count might not match the total credit count of its TCs in the following cases:
  - A TC is defined as GSP — when a GSP is selected, the BWG credits are decremented but no TC is deducted. Therefore, the BWG credit count would be lower that the credit count of its TCs.
  - Max credits during refill — If a TC reaches its max credits value during refill, then some credits are lost for that TC. However, the BWG for that TC is provided with the full refill count. Therefore, the BWG credit count would be higher that the credit count of its TCs.
- One bit per TC arbiter governs the recycle mode of the WSP algorithm:
  - 0: No Recycling — At the end of each full arbitration cycle all TC's are refilled with their TC.Refill up to their TC.MaxCredits values. All BWG.Credit are loaded by the sum of the BWG TC.Credit.
  - 1: Recycle — TC credits for TC's that have reached their maximum are recycled to other TC's of the BWG. The operation is calculated based on the BWG.Credit and the TC.Credits after their refill. The difference between them is the BWG.Recycle value.

    Positive BWG.Recycle - The recycle algorithm adds credits from the BWG.Recycle to the TC.Credits starting from the highest priority TC in the BWG down, considering the TC.MaxCredit, until BWG.Recycle is zero.

    Negative BWG.Recycle - The recycle algorithm subtracts credits from the BWG.Recycle to the TC.Credits starting from the lowest priority TC in the BWG up, until BWG.Recycle is zero.

A separate set of configuration parameters exists for each of the three TC arbiters as listed in Table 7-56.

Table 7-56 Configuration Parameters for the Tx and Rx TC Arbiters

Parameter	Tx Packet Arbiter	Tx Descriptor Arbiter	Rx Packet Arbiter
	RTTPCS	RTTDCS	RTRPCS
RR / WSP mode	TPPAC	TDPAC	RAC
Recycle mode	TPRM	TDRM	RRM

7.7.2.4.4 Tx TC WSP Arbiter Algorithm

The Transmit Packet Plane Arbitration Control (TPPAC) bit in the RTTPCS registers determines the scheduler type (RR or WSP).

Strict Priority — The strict-priority aspect of the TC WSP arbiter resides in the fact that once a TC has been granted for a data read request or for transmission, the highest priority TCs are checked (again) in a strict-priority order, starting from TC7, even if the granted TC still has credits for another data read request or transmission.

Note: The descriptor plane arbiter can't issue a data read request unless there is an unused request for data. Therefore the arbiter stalls in its current state each time there are no data read requests available. The next arbitration decision is only done once there is at least one free data read request.

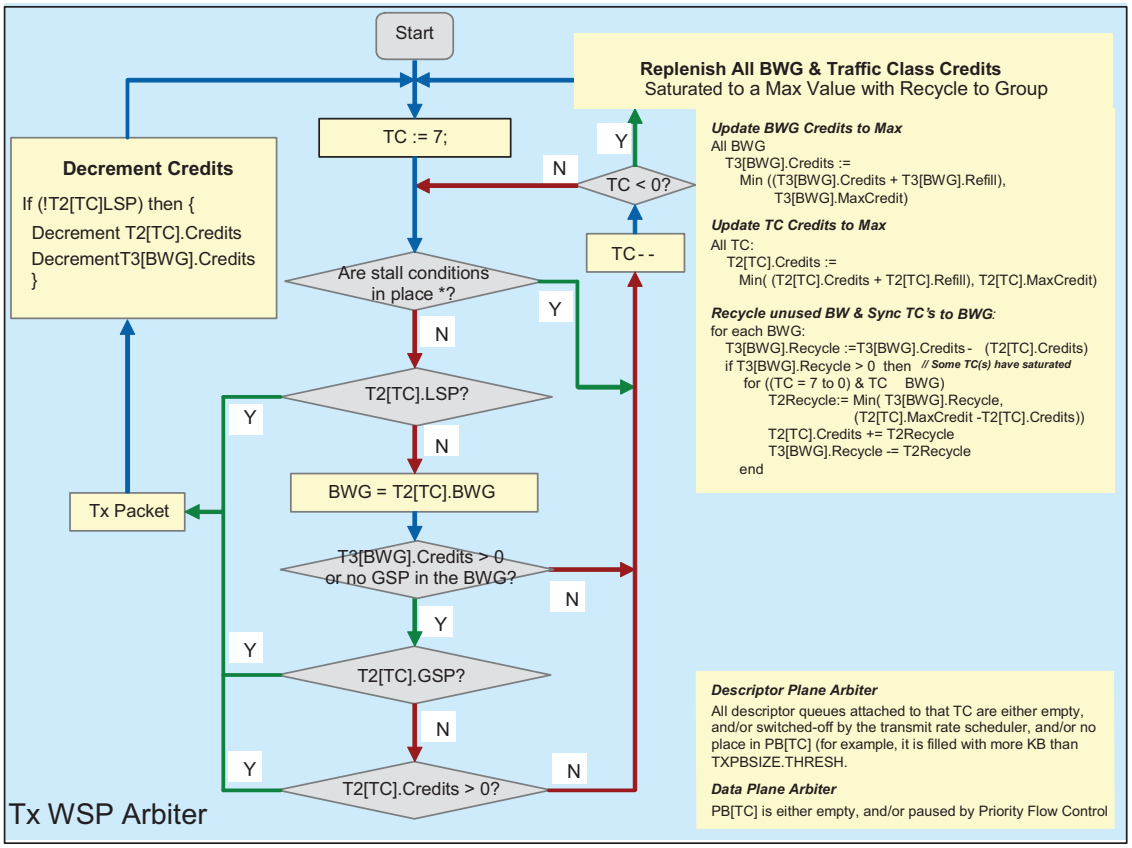


Figure 7-34 Tx TC WSP Arbiters Operation

7.7.3 Receive-Side Capabilities

7.7.3.1 User Priority to Traffic Class Mapping

To enable different TC support for incoming packets and proper behavior per TC, the 82599's receive packet buffer is segmented into several packet buffers. The 82599 supports the following configurations of packet buffer segmentation:

- DCB disabled — Single buffer of 512 KB
- DCB enabled with 4 TCs — 4 buffers, 128 KB each
- DCB enabled with 8 TCs — 8 buffers, 64 KB each
- DCB enabled with 8 TCs — 8 buffers, buffers 3:0 are 80 KB and buffers 7:4 are 48 KB

Incoming packets are transferred from the packet buffers into data buffers in system memory. Data buffers are arranged around descriptor rings described in Section 7.1.9. Each descriptor queue is assigned dynamically to a given TC (and therefore to a packet buffer) as described in Section 7.1.2.

Configuration registers:

- The size of each buffer is defined by the RXPBSIZE[0-7] registers. Note that it is possible to configure the buffers at 1 KB granularity
- A received packet is assigned by the 82599 to a TC, and is thus routed to the corresponding Rx packet buffer according to its *User Priority* field in the 802.1Q tag and according to a UP to TC mapping table loaded into the RTRUP2TC register.

Caution: Different UP to TC mappings can be loaded in each direction Tx and Rx, as per RTTUP2TC and RTRUP2TC registers, respectively. But in such a case, when a packet is looped back by the internal VM to VM switch, it is routed to the Rx packet buffer that corresponds to the TC that was used in Tx.

7.7.3.2 Rx PB Weighted Strict Priority Arbiter

The 82599's Rx arbiter determines the order in which packets are written from the different packet buffers into system memory. Note that each packet buffer by itself is drained in the order packets arrived, as long as it deals with packets destined to the same Rx queue.

The arbitration algorithm between the receive packet buffers is WSP, similar to the TC scheme on the transmit side. Motivation for this scheme is as follows:

1. The major consideration is to prevent any delay in delivery of high-priority traffic.
2. Allocation of credits controls the bandwidth allocated to the different packet buffers.
3. A secondary mean of bandwidth allocation is the priority flow control. By altering the flow control high watermark, the 82599 can effectively (if coarsely) regulate bandwidth allocation to types of traffic.

Table 7-57 (T4) defines the TCs and their bandwidth allocation. The following elements are defined in this table:

Table 7-57 Bandwidth Allocation to Traffic Classes and Bandwidth Groups

T4: Packet Buffer Bandwidth Allocation Within a BWG						
PB	**BWG**	**PB Refill**	**PB Max Credits**	**LSP**	**GSP**	**PB Min Credits (according to GSP 0/1)**
0	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
1	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
2	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
3	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
4	2	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
5	3	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
6	1	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB
7	0	64 bytes to 32 KB	64 bytes to 256 KB	0/1	0/1	-MSS / -2,048 KB

† Due to a pipelined implementation, the PB credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

†† All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

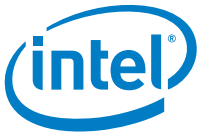


Table 7-58 (T5) defines the hierarchy of BWG similarly to T4 defining the hierarchy within the BWG's. T4 implementation should include eight rows.

Table 7-58 Packet Buffer Allocations to BWGs (with Example)

T5: Line Bandwidth Allocation to Traffic BWGs				
BWG	**BWG Credit Refill**	**BWG MaxCredit**	**BWG Credit**	**Description**
0	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048KB	IPC
1	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048KB	SAN
2	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048KB	LAN
3	= Σ[TC] Credit Refill	= Σ[TC]. MaxCredit	-/+ 2,048KB	MGMT.
-				
-				
-				
7				

The attributes of the Rx packet arbiter are described in Section 7.7.2.4.2.

7.7.3.2.1 Rx TC Arbitration Configurations

Table 7-59 lists the register fields that control the Rx arbiter.

Table 7-59 Registers Allocation for DCB Rx Arbiters

Attribute	Rx Packet Arbiter
PB Control registers	RTRPT4C
PB Status registers	RTRPT4S
BWG	BWG
PB credit refill	CRQ
PB MaxCredit	MCL
LSP	LSP
GSP	GSP
PB credits	CCC

Configuration parameters for the Rx packet arbiter are defined and listed in Section 7.7.2.4.1.

7.7.3.2.2 Rx TC WSP Arbiter Algorithm

The Rx packet arbiter operates in RR or WSP mode, configured through the *RAC* bit in the RTRPCS register. The WSP arbiter operation is described as follows.

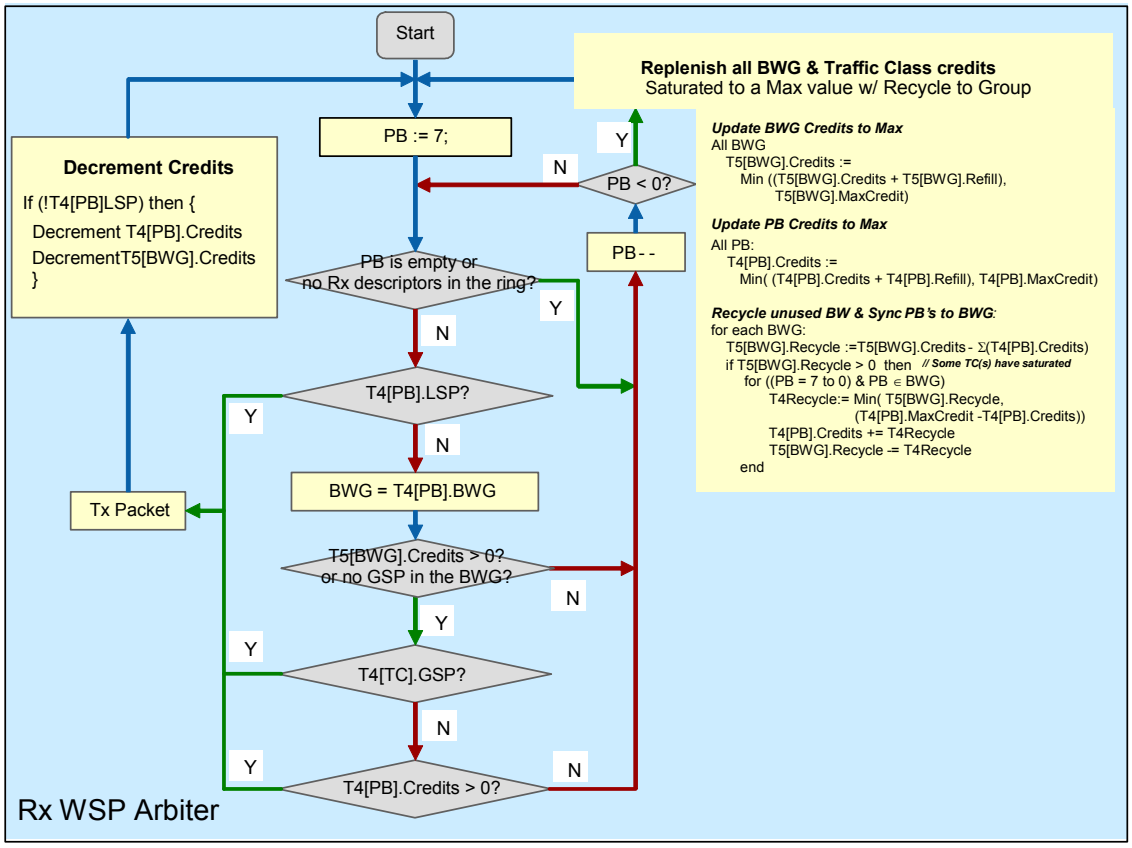


Figure 7-35 Rx Packet WSP Arbiter Operation