6. Media access control data path

Editors' Notes: *To be removed prior to final publication.*

*R*eferences:
None

Definitions:
None.

Abbreviations:

PTQ	Primary transit queue.
STQ	Secondary transit queue.

Revision History:

Draft 0.3, June 2002	Adopted by TF and introduced into P802.17.

This clause describes per-ringlet behavior, unless explicitly mentioned otherwise.

6.1 Flow-control overview

Editors' Notes (DVJ): *To be removed prior to final publication.*

The following acronyms should be updated in the central clause:

PTQ: primary transit queue.
STQ: secondary transit queue.

6.1.1 Traffic classes

Client data is classified into 3 traffic classes, as listed in table 6.1. Class-A traffic is provisioned with a committed information rate (CIR), and provides the lowest MAC delay and jitter[1] bounds. Class-B traffic is provisioned with a CIR, and provides bounded MAC delay and jitter for that traffic within the profile of the CIR. Class-B traffic beyond the provisioned CIR is referred to as excess information rate (EIR) class-B traffic. Class-C traffic is not provisioned. EIR class-B and class-C traffic is marked as out of profile by the MAC, and provide no MAC delay or jitter bounds.

Editors' Notes: *To be removed prior to final publication.*

*The exact jitter bounds have not been calculated. Class-A jitter is on the order of N * MTU, where N is the number of stations on the ring. Class-B jitter is on the order of RTT. More precise calculations are needed.*

Table 6.1—Service ~~Classes~~classes

Name	~~Bandwdith~~Bandwidth	Jitter	Row	Example of use
Class-A	provisioned	small	6.1.1	Real time
Class-B	provisioned	bounded	6.1.2	Near real time
Class-C	opportunistic	unbounded	6.1.3	Best effort network traffic

[1]MAC delay and jitter are defined to be measured from the point where a packet arrives at the head of a source client add queue until it arrives at the tail of a destination client receive queue.

The class-C traffic is opportunistic rather than provisioned, in that only unprovisioned or unused reclaimable provisioned class-A/B bandwidths are available. As such, the class-C traffic provides no minimum-bandwidth or maximum-jitter guarantees. A weighted fairness algorithm is used to partition class-C traffic among contending stations.

Internal to the MAC, the class-A bandwidth is partitioned into two subclasses: subclass-A0 and subclass-A1. The MAC's client requests ~~claas~~class-A traffic, not one of the internal subclasses. The MAC is provisioned for a total class-A amount, from which it determines how much is subclass-A0 and how much is subclass-A1, based on ring circumference and STQ size. The MAC advertises a class-A provisioning equal to its internal subclass-A0 amount.

6.1.2 MAC data paths

Each attach point has one or two transit queues, for saving pass-through traffic that arrives during this stations's transmissions, as shown in ~~Figure 6.7~~Figure 6.1. There are two types of MAC transit queueing designs: mono-queue and dual-queue (see 6.4 and 6.5). The mono-queue design places all pass-through into a primary transit queue (PTQ); the dual-queue design places class-A traffic into a higher-precedence primary transit queue (PTQ) and other traffic into a lower-precedence secondary transit queue (STQ).

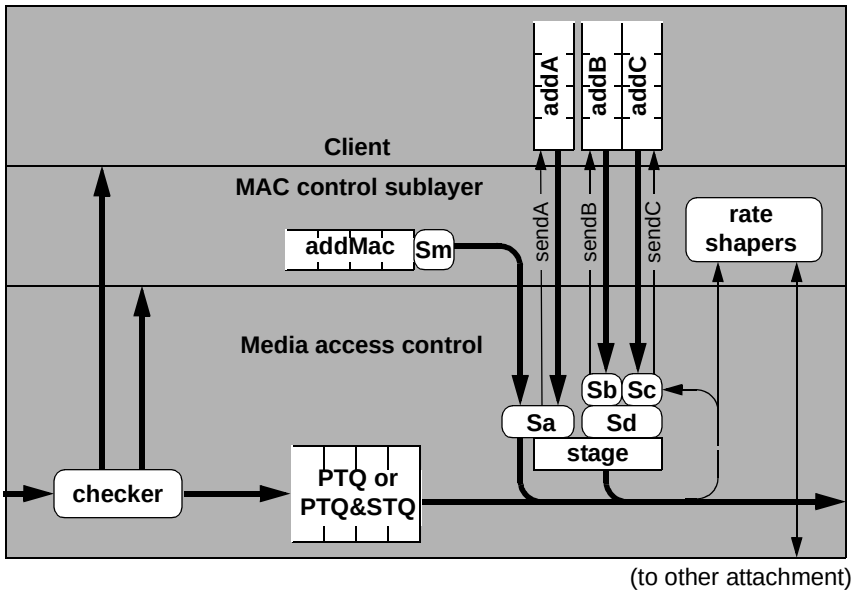

Figure 6.1—MAC data paths

The client has the option of labeling its frames as class-A, class-B, or class-C. The class-A, class-B, and class-C traffic flow through the Sa, Sb, and Sc shapers, respectively. Additionally, excess class-B traffic is marked out of profile by the MAC and also uses the Sc shaper.

Accepted client traffic is expected to be placed into a stage queue. Sufficient stage-queue storage ensures full-rate transmissions, despite the latencies inherent in signaling rate-limiting flow-control information across the MAC-to-client interface.

Packets from the MAC-control send queue are rate-limited by the ***Sm*** (shaper of MAC), to avoid disruption of provisioned class-A traffic due to unexpected bursts of control traffic. All class-A traffic is rate-limited by the ***Sa*** (shaper of class-A), to avoid having the client exceed its class-A provisioned rates. MAC-control and client traffic both flow through the same shaper, although the shaper effectively only throttles the client's cumulative class-A traffic.

The client's class-B traffic is rate-limited by the ***Sb*** (shaper of class-B), to constrain the client within its class-B provisioned rates. Similarly, the client's class-C traffic is rate-limited by the ***Sc*** (shaper of class-C), to constrain the client within its weighted fair-share use of the residual (unprovisioned or unused provisioned) bandwidth.

The client's class-B and class-C traffic are both shaped by the ***Sd*** (shaper for downstream), to constrain the client for sustaining downstream provisioned class-A rates. Thus, in profile class-B flows are only possible when allowed by shaper *Sb* and *Sd*; similarly, excess class-B and class-C flows are only possible when allowed by shapers *Sc* and *Sd*.

Flow control indications are generated by the MAC, to selectively restrict the ~~tranffic~~ traffic flows from the client. Simple ***sendA*** and ***sendB*** go/no-go indications provide for throttling class-A and class-B traffic, but ***sendC*** communicates a maximum hop-count distance so that class-C traffic can be selectively throttled based on the distance to its target.

Editors' Notes: *To be removed prior to final publication.*

Previously described stopA, stopB, and stopC are equivalent to sendA==0, sendB==0, and sendC==0 respectively.

~~NOTE—sendX is logically equivalent to NOT stopX.~~

6.1.3 Reclamation

Provisioned bandwidth can be reused, or reclaimed, by a lower priority class whenever the reclamation does not effect the jitter bounds of the higher priority class(es) on the local station or on any other station on the ring.

NOTE—Traffic can be sent one hop when there is no traffic in the PTQ, regardless of the provisioning on the link. This is because the maximum delay for any PTQ traffic coming into the local station is 1 MTU, and because the added frame gets stripped at the neighboring station, creating an idle space for any PTQ traffic that needs to enter at the neighboring station.

6.2 Bandwidth provisioning

Editors' Notes: *To be removed prior to final publication.*

This subclause contains introductory material and may therefore be moved to the introduction clause when this document is revised in the future.

The means of provisioning the class-A and class-B traffic, assuring the provisioning is consistent across the ring, and communicating the provisioned amounts to the stations controlling the links over which the traffic transits is left to the OAM&P section drafters.

Note that the communication of provisioning information must be phased so as to not to create a case where transitioning provisioning levels temporarily cause cumulative provisions to exceed the link capacity.

Whether bandwidth is provisioned globally uniform or differently for each link based on spatial awareness of the traffic being provisioned is an implementation decision.

Each station shall have default values for provisioned bandwidths to enable it to plug and play without management configuration.

There are two forms of traffic provisioning, uniform and spatial. Uniform provisioning uses a single rate per class for the entire ring. Spatial provisioning uses independent rates per class for each link. Uniform provisioning makes no distinction between traffic flows based on their hop-count distance. Spatial provisioning differentiates traffic amounts on a per hop-count basis.

Class-A traffic is divided into subclasses, subclass-A0 and subclass-A1. The subclass-A1 is more efficient, but relies on sufficient secondary transit queue depth to buffer class-B and class-C traffic while a congested station is signaling upstream stations to decrease excess traffic. The amount of supportable subclass-A1 traffic in any station is proportional to the size of the secondary transit queue in that station.

All flow-control protocols assume that each station is aware of its class profile as well as the ringlet's cumulative class profile. The simple flow-control protocols (that assume flat class profiles) and the sophisticated flow-control protocols (that allow hop-count dependent class profiles) are fully interoperable.

6.2.1 Mono-queue uniform provisioning

A simple mono-queue station has provisioned levels of class-A and class-B traffic, as illustrated in Figure 6.1. Within each figure cell, a bandwidth profile is illustrated above the physical ring illustration. In this simplest of provisioning examples, provisioning levels are uniform across the ring, not hop-count dependent. Uniform provisioning makes no distinction between class-A traffic (cell 1) sent from W-to-X and class-A traffic sent from W-to-Z. Similarly, uniform provisioning makes no distinction between class-B traffic (cell 2) traffic sent from W-to-Y and class-B traffic sent from W-to-Z.

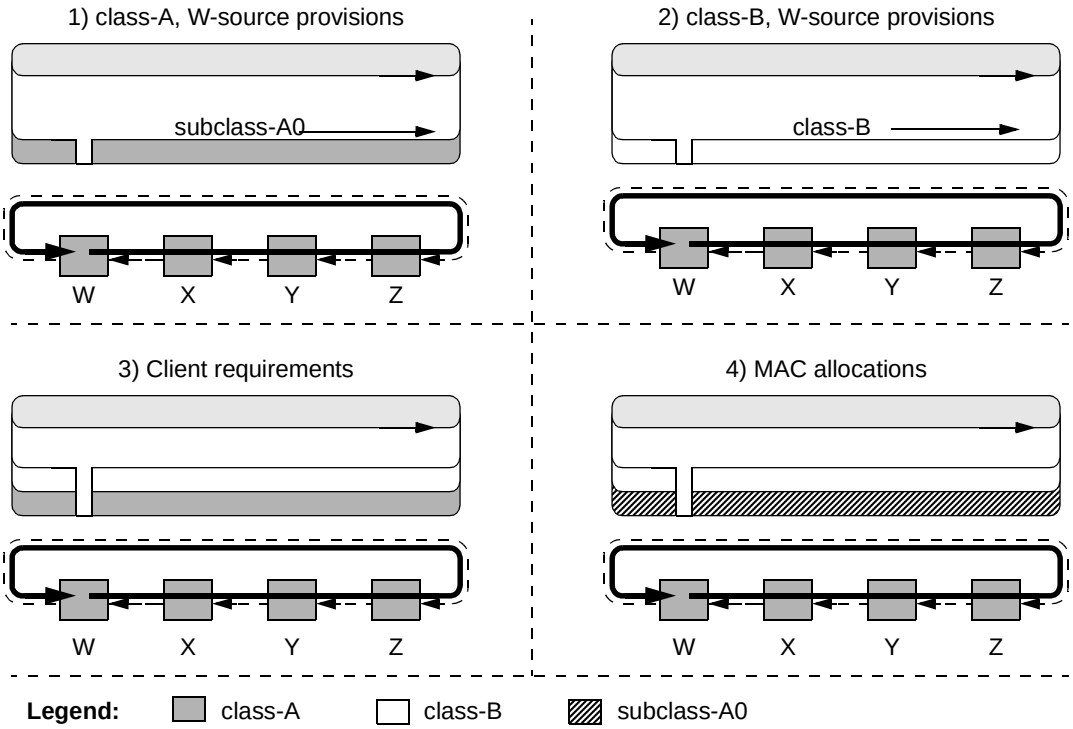

Figure 6.1—Mono-queue uniform provisioning

For each station, the cumulative profiles form a cumulative bandwidth profile (cell 3) of client-visible bandwidth allocations. Within a mono-queue MAC, the bandwidth advertised for the class-A service (cell 4) consists of the entire class-A provisioned bandwidth, since a second transit queue is necessary to support advertisement of a more efficient subclass-A1 advertisement.

6.2.2 Dual-queue uniform provisioning

A more storage-rich dual-queue station has advertised provisioned levels of subclass-A0, subclass-A1, and class-B traffic, as illustrated in Figure 6.2. Uniform provisioning makes no distinction between class-A traffic (cell 1) sent from W-to-X and class-A traffic sent from W-to-Z. Similarly, the provisioning (cell 2) of class-B traffic makes no distinction between traffic sent from W-to-Y and traffic sent from W-to-Z.

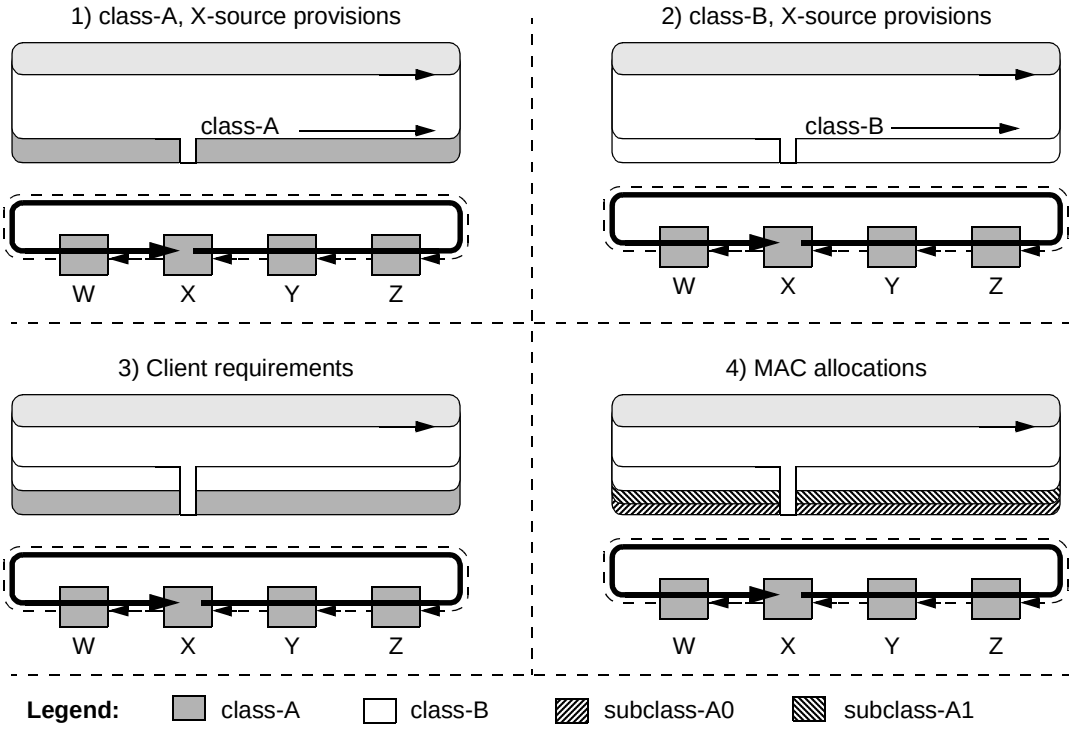

Figure 6.2—Dual-queue uniform provisioning

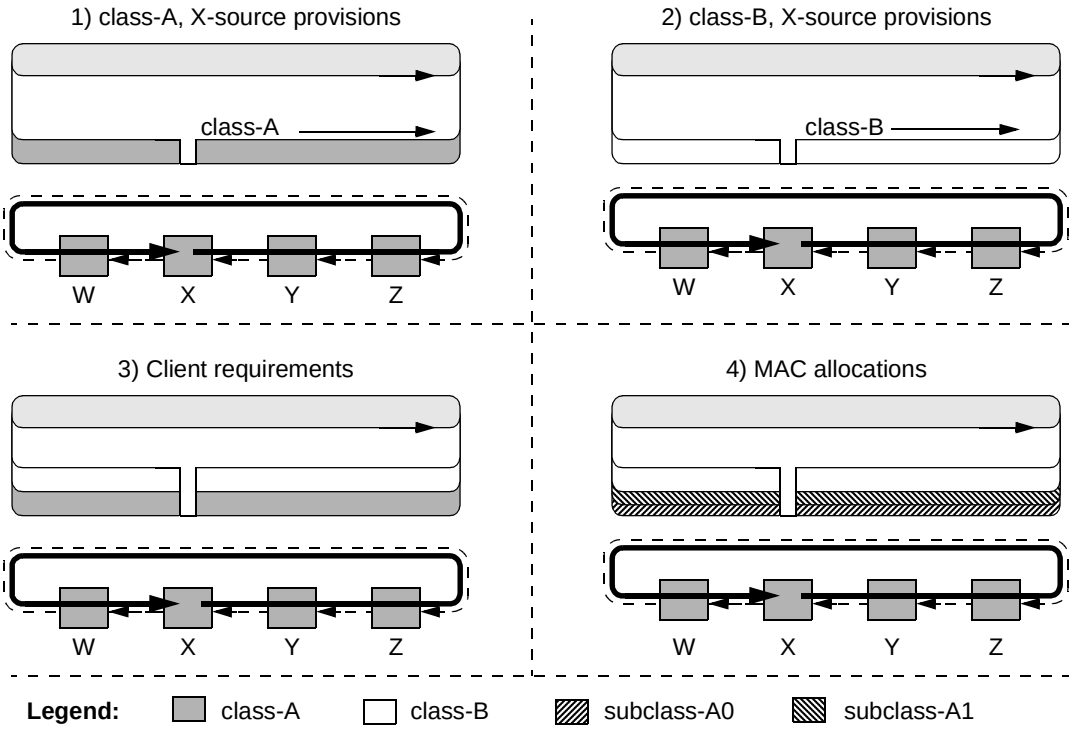

Figure 6.3—Dual-queue uniform provisioning

Editors' Notes: *To be removed prior to final publication.*

These figures need to be reviewed to assure they match the text, or vice versa.

For each station, the sum of class profiles forms a cumulative bandwidth profile (cell 3) of client-visible class-A and class-B bandwidth allocations. The class-A service (cell 4) consists of the allocation of subclass-A0 and class-A1 bandwidths, where the levels of supportable subclass-A1 bandwidths are proportional to the size of the secondary transit queue.

6.2.3 Mono-queue spatial provisioning (informative)

Each station has provisioned levels of class-A and class-B traffic, as illustrated in Figure 6.4. Spatial provisioning makes a distinction between class-A traffic sent (cell 1) from Y-to-X and class-A traffic (cell 2) sent from Y-to-W. Similarly, spatial provisioning makes a distinction between class-B traffic (cell 3) sent from Y-to-W and class-B traffic (cell 4) sent from Y-to-Z.

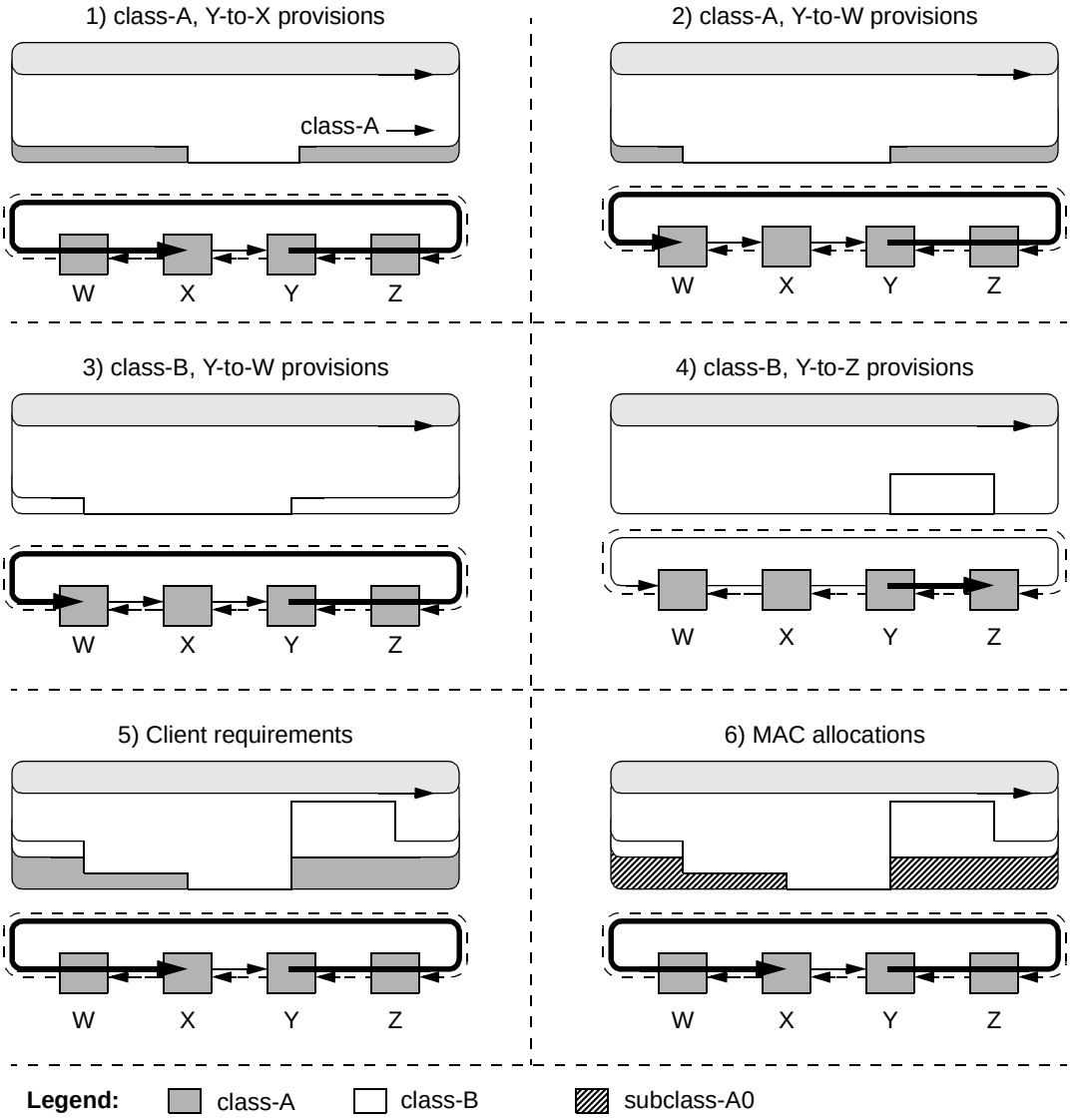

Figure 6.4—Mono-queue spatial provisioning

For each station, the sum of class profiles forms a cumulative bandwidth profile (cell 3) of client-visible class-A and class-B bandwidth allocations. The class-A service (cell 4) consists of only subclass-A0 bandwidth, since a second transit queue is necessary to support advertisement of a more efficient subclass-A1 advertisement.

6.2.4 Dual-queue spatial provisioning (informative)

Each station has provisioned levels of class-A and class-B traffic, as illustrated in Figure 6.5. Spatial provisioning of class-A traffic makes a distinction between (cell 1) traffic sent from Z-to-Y and (cell 2) traffic sent from Z-to-X. Similarly, spatial provisioning makes a distinction between class-B traffic (cell 3) sent from Z-to-X and class-B traffic (cell 4) sent from Z-to-W

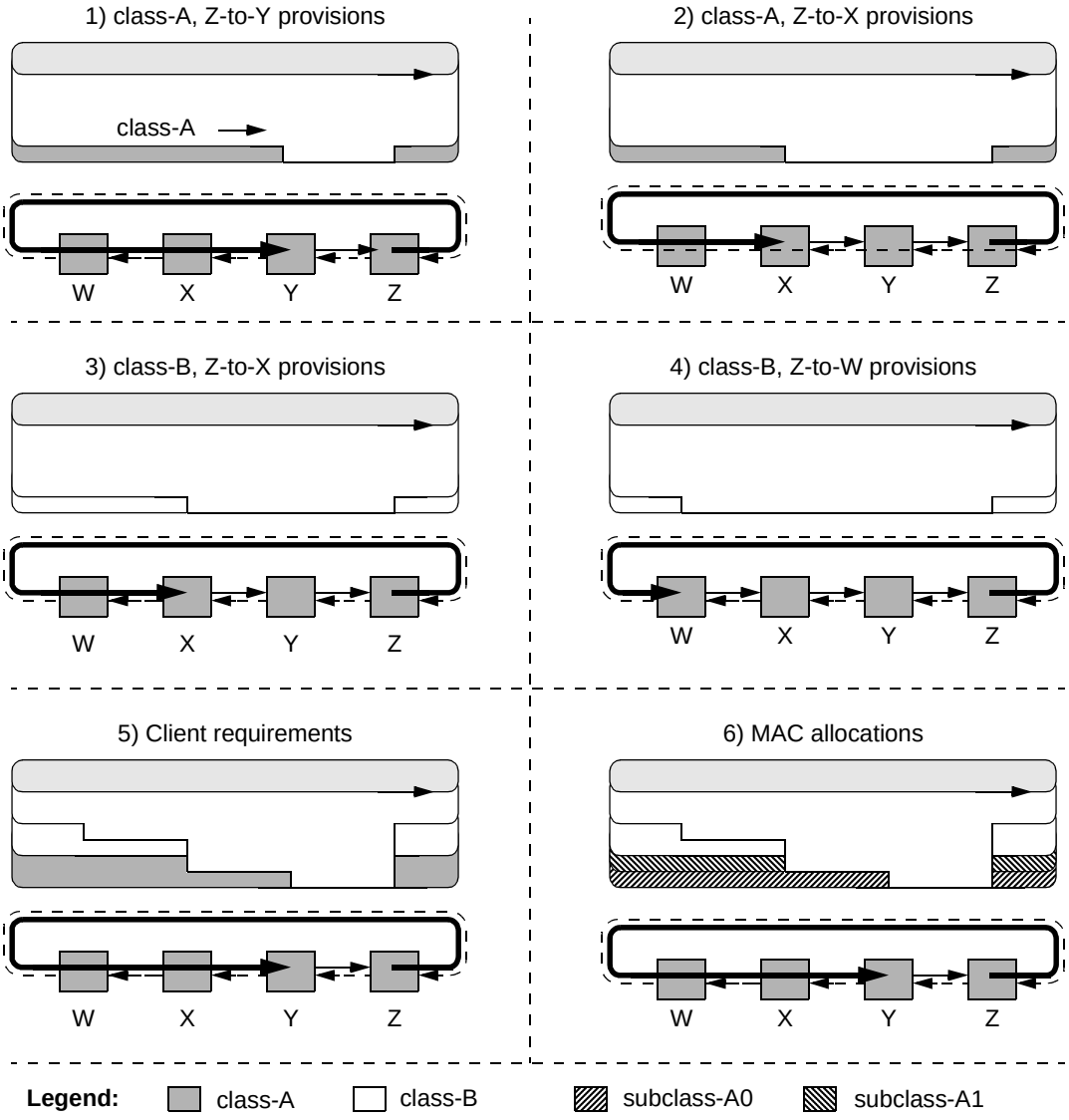

Figure 6.5—Dual-queue spatial provisioning

For each station, the sum of class profiles forms a cumulative bandwidth profile (cell 3) of client-visible class-A and class-B bandwidth allocations. The class-A service (cell 4) consists of subclass-A0 and subclass-A1 bandwidths, where the level of supportable subclass-A1 bandwidth is proportional to the size of the secondary transit queue.

6.2.5 Cumulative ringlet provisioning (informative)

Each station has its own (cells 1, 2, 3, and 4) cumulative class profile, as illustrated in Figure 6.6. These can be combined (cell 5) into a ringlet class profile. Within consistent ringlet profiles, the sum of subclass-A0, subclass-A1, and class-B profiles (*provisioned* in cell 5) shall be less than any individual link capacity.

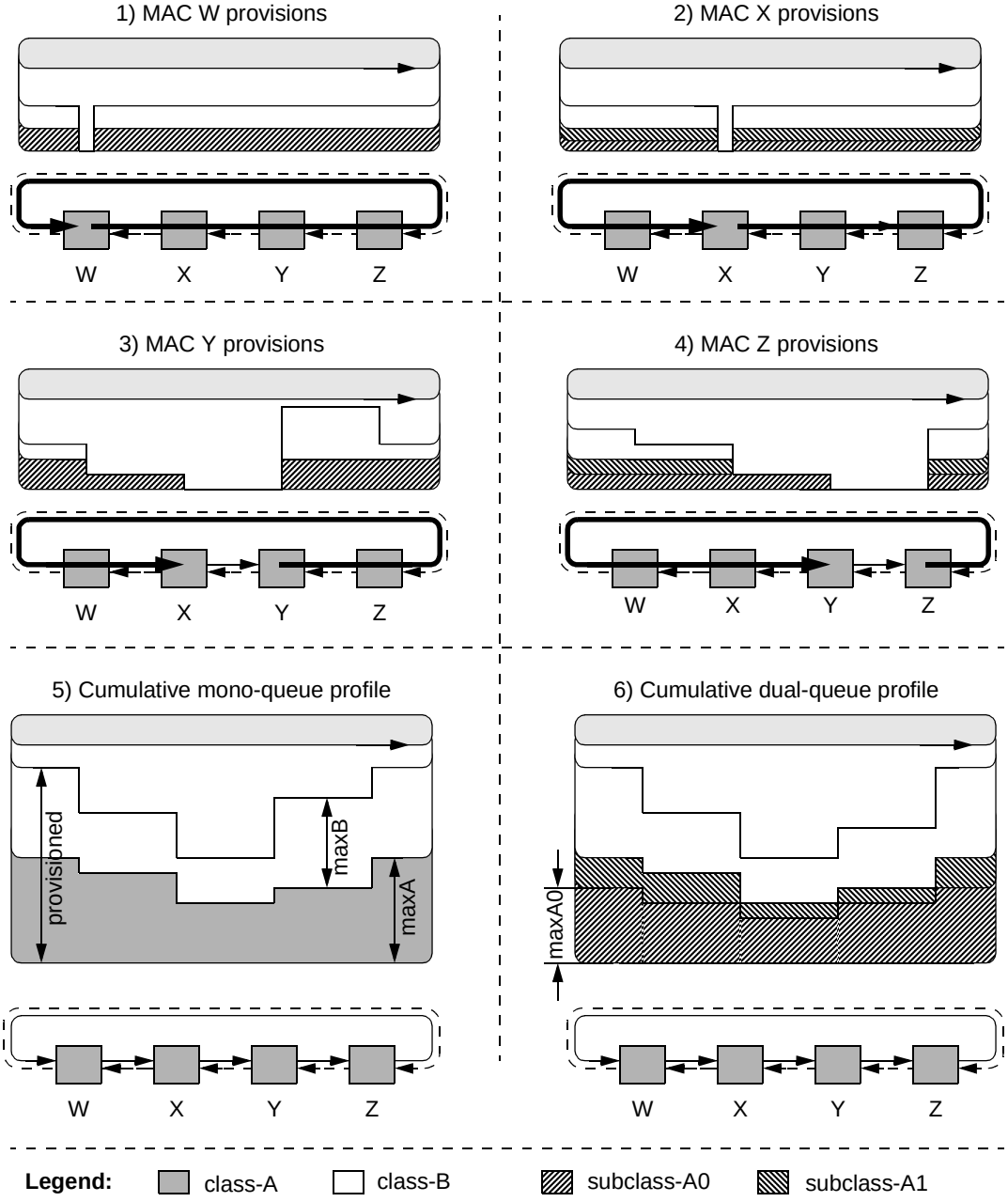

Figure 6.6—Cumulative station provisions

For scalar policing, the cumulative class profile yields *maxA* and *maxA0*, the worst-case provisioned class-A and subclass-A0 segments respectively. For vector policing, the cumulative class profile provides *rateA[n]* and *rateA0[n]*, the provisioned class-A and subclass-A0 levels on segment *n*, respectively. This information

is used to rate-limit each station's class-B and class-C transmissions, with the intent of sustaining downstream class-A transmissions.

During provisioning, the cumulative class profile also yields *maxB*, the worst-case provisioned class-B segment. To ensure interoperability, the sum of *maxA* and *maxB* shall be less than the link capacity.

NOTE—The value of *rateA+rateB* equals the maximum value of *rateA[n]+rateB[m]*, measured on all links *n* and *m*. The requirement for this sum to be less than the link capacity is more restrictive than the physical mandated restriction that *rate[n]+rateB[n]* be less than the link capacity on any link *n*.

6.3 Rate control

6.3.1 Add queue rate policing

Packets from the MAC Control add queue, class-A add queue, and class-B add queue are rate controlled, via the ***sendA*** and ***sendB*** signals, to the class-A and class-B rates provisioned for the station. Fairness ~~eligbile~~ eligible add traffic (Class-C and EIR class-B traffic) is rate controlled, via the ***sendC*** signal, to 2 rates. It is always controlled to a static rate configured for the station. In addition, any class-C traffic passing the congestion point reported in the Type 1 fairness message is also rate controlled to the lesser of the static rate and the dynamic rate provided by the RPR-fa algorithm. The ***sendC*** signal can be used to indicate which of these rate limitations is in effect by including the distance to the congestion point. No congestion point is indicated by a distance of the ring size. In addition to the above, all classes of traffic are effectively rate limited by the transmission selection algorithms described in 6.4.2 and 6.5.2.

6.3.2 Mono-queue rate-shaping

In a mono-queue implementation, the PTQ output is not shaped.

6.3.3 Dual-queue rate-shaping

In a dual-queue implementation, the PTQ output does not need to be shaped. The total outgoing (add plus transit) sum of (subclass-A1 + class-B + class-C) traffic output shall be shaped to meet the class-A requirements of downstream stations.

6.3.4 MAC shapers

Although multiple shapers are used within this standard, the behaviors of all shapers can be characterized by a common algorithm with application-specific parameters. The shaper's *dSize*, *uSize*, and *rate* parameters effect the credit adjustments and limits, as illustrated in Figure 6.7. The *dSize* and *uSize* values typically represent sizes of a transmitted frame and update intervals respectively.

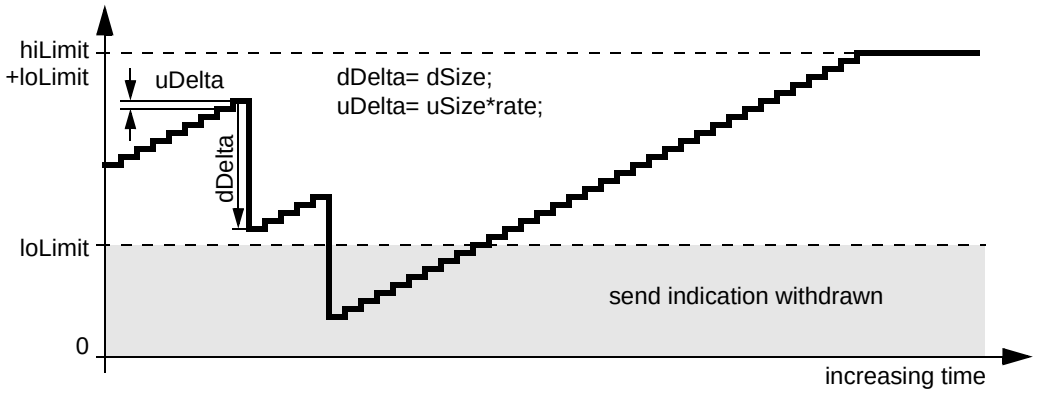

Figure 6.7—Credit adjustments over time

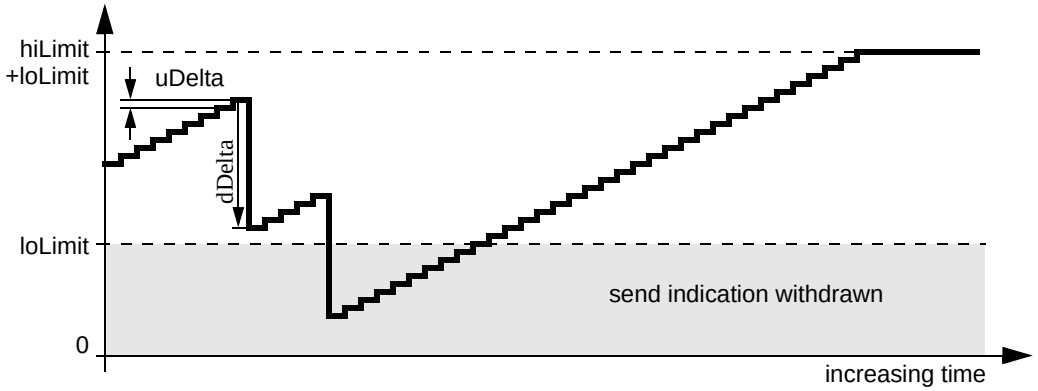

Figure 6.8—Credit adjustments over time

Crossing below the *loLimit* threshold typically generates a rate-limiting indication, so that offered traffic can stop before reaching the lesser MTU limit, where excessive transmissions are rejected.

The *hiLimit* value limits the positive credits, to avoid overflow and to bound the burst traffic after inactivity intervals. The term *hiLimits* is used to represent the sum of *hiLimit* and *loLimit* values.

6.4 Mono-queue MAC design

A mono-queue MAC uses one queue, the primary transit queue (PTQ), for all transit traffic. The PTQ is at least 2 MTUs (in order to allow almost immediate access for control frames).

6.4.1 Mono-queue MAC data paths

To be able to detect when to transmit and receive packets from the ring, a mono-queue MAC makes use of only one transit queue, as shown in Figure 6.9. The PTQ has the behavior of a small FIFO.

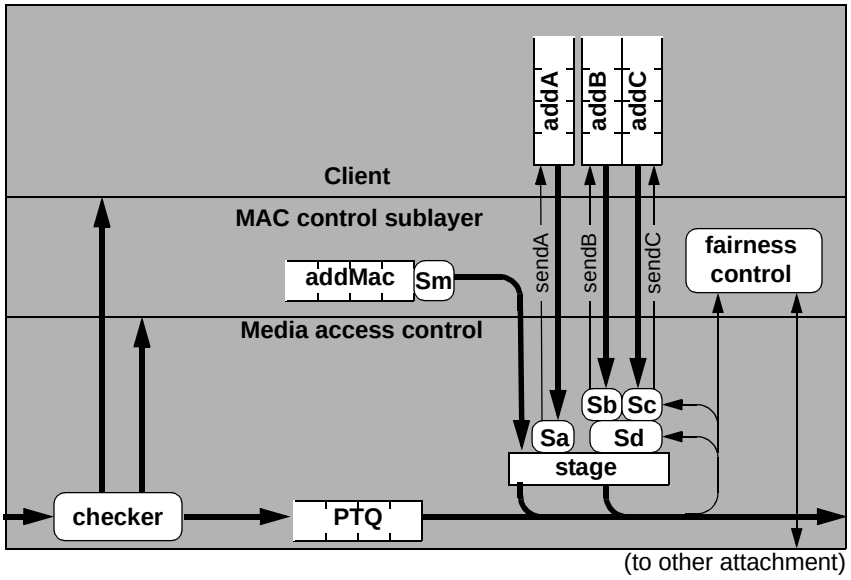

Figure 6.9—Mono-queue MAC data path

Packets from the MAC-control send queue are rate-limited by shaper ***Sm*** (shaper of MAC). All class-A traffic is rate-limited by shaper ***Sa*** (shaper of class-A).

This standard does not define how to implement the PTQ. However, to meet ordering expectations, a FIFO ordering shall be maintained when entries pass through the PTQ.

6.4.2 Mono-queue transmit selections

The behavior of a mono-queue MAC is described by its transmit-selection protocols (described in this subclause) and shaping functions. The transmit-selection protocol and shaping functions are largely independent, but some coupling (via the internal shaper provided *sendM* and *sendA* signals) is required to ensure conformance of MAC-supplied class-A control frame transmissions.

A mono-queue MAC can transmit data packets from three possible internal queues, as illustrated in Figure 6.11. An exact definition of the mono-queue transmit-frame selection sequence is specified in table 6.2, where the rows are evaluated in top-to-bottom order. The intent is to always empty the primary

transit queue (PTQ) before frame transmissions, to avoid queue overflow conditions associated with

enqueueing additional incoming frames during frame transmissions.

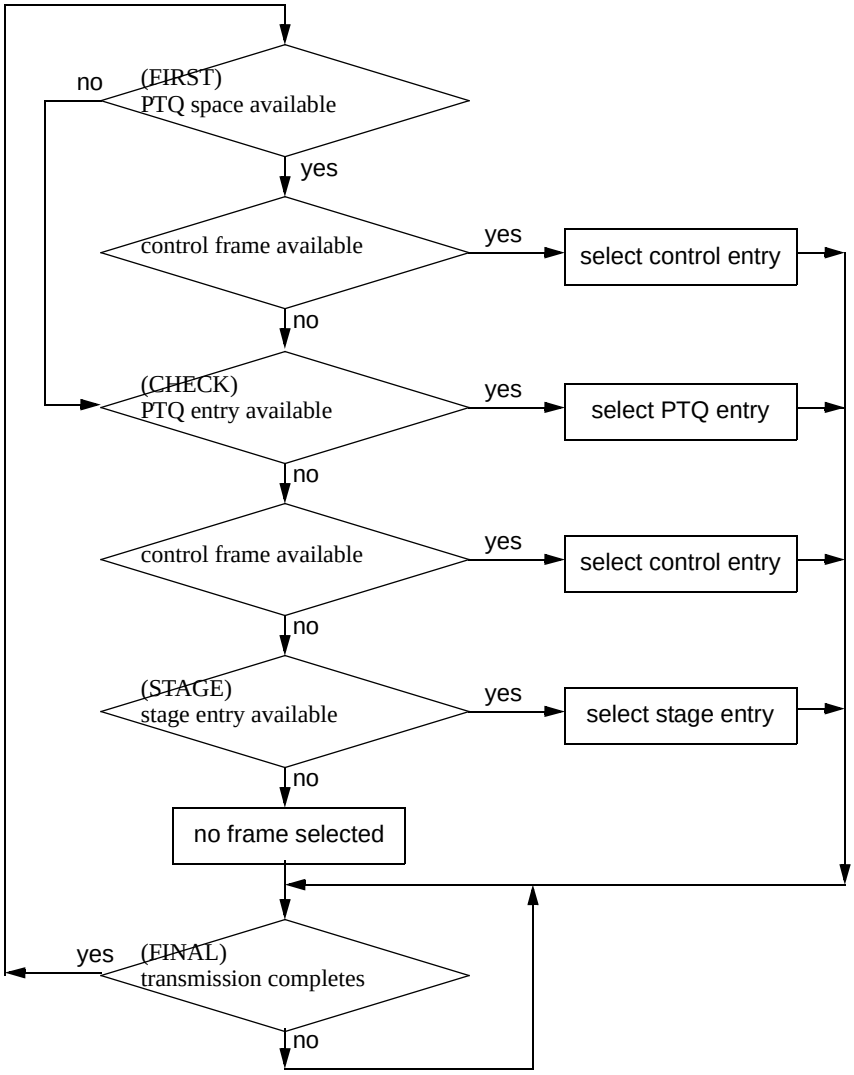

Figure 6.10—Mono-queue transmit-frame selection

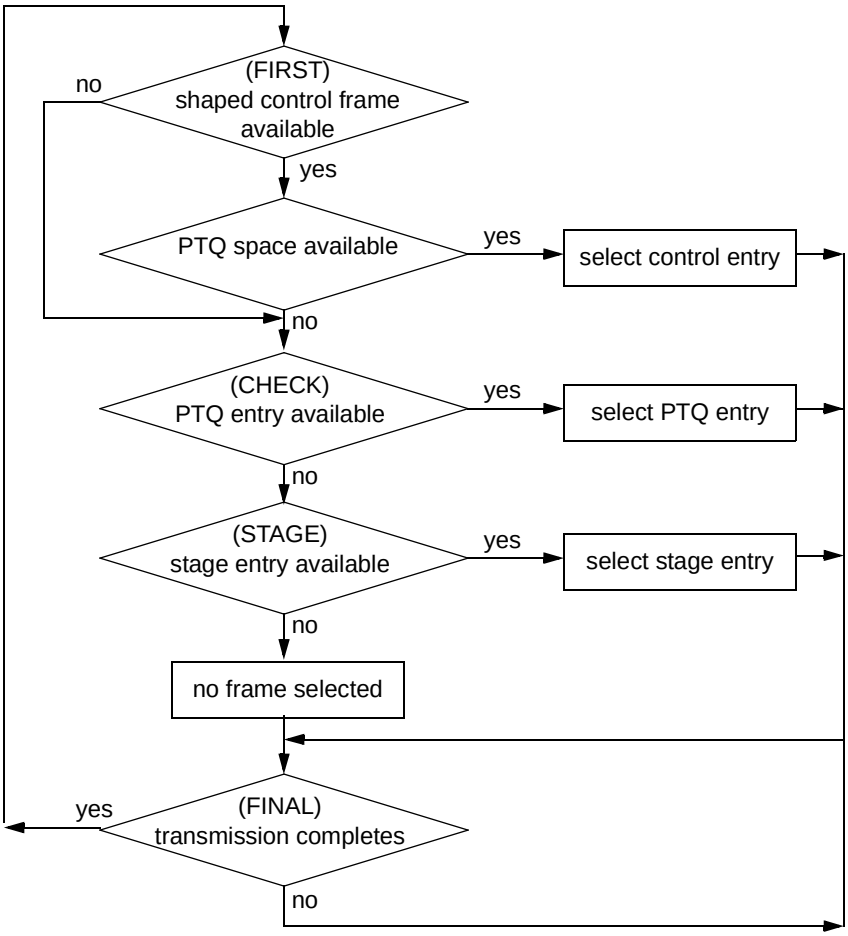

Figure 6.11—Mono-queue transmit-frame selection

Table 6.1—Mono-queue transmit-frame selection

Last state		Row	Next state	
state	**condition**		**selection**	**state**
FIRST	sizeOfMacControl > spaceInPTQ	6.1.1	—	CHECK
	passM == 0	6.1.2		
	—	6.1.3	MAC control queue	FINAL
CHECK	CUT_THOUGH && headerInPTQ	6.1.4	primary transit queue	FINAL
	STORE_FORWARD && frameInPTQ	6.1.5		
	frameInMacControl == 0	6.1.6	—	STAGE
	passM == 0	6.1.7	—	
	passA == 0	6.1.8	—	
	—	6.1.9	MAC control queue	FINAL
STAGE	CUT_THOUGH && headerInStage	6.1.10	stage queue	FINAL
	STORE_FORWARD && frameInStage	6.1.11		
	—	6.1.12	no frame selected	
FINAL	selected transfer completes	6.1.13	—	FIRST
	—	6.1.14	—	FINAL

Row 6.2.1: In the absence of MAC-control frames, other transmission sources are checked.
Row 6.2.2: In the absence of MAC-control transmission credits, other transmission sources are checked.
Row 6.2.3: The queued MAC exceeds available PTQ space, other transmission sources are checked.
Row 6.2.4: The MAC-control transmissions have precedence over client transmissions.

Row 6.2.5: (Representing the cut-through option)
The primary transit queue is selected when a header is available, to avoid queue overflows.
Row 6.2.6: (Representing the store-and-forward option)
The primary transit queue is selected when a frame is available, to avoid queue overflows.
Row 6.2.7: In the absence of PTQ transit frames, the client-supplied stage-queue is checked.

Row ~~6.2.1:~~ ~~The size of the queued MAC control frame shall be less than the available PTQ space.~~
~~Row 6.2.2:~~ ~~In the absence of MAC-control transmission credits, other transmission sources are checked.~~
~~Row 6.2.4:~~ ~~The small MAC-control transmissions have precedence over client transmissions.~~
~~Row 6.2.5:~~ ~~(Representing the cut-through option)~~
~~The primary transit queue is selected when a header is available, to avoid queue overflows.~~

Table 6.2—Mono-queue transmit-frame selection

Last state		Row	Next state	
state	**condition**		**selection**	**state**
FIRST	frameInMacControl == 0	6.2.1	—	CHECK
	passM == 0	6.2.2		
	sizeOfMacControl > spaceInPTQ	6.2.3		
	—	6.2.4	MAC control queue	FINAL
CHECK	CUT_THOUGH && headerInPTQ	6.2.5	primary transit queue	FINAL
	STORE_FORWARD && frameInPTQ	6.2.6		
	—	6.2.7	—	STAGE
STAGE	CUT_THOUGH && headerInStage	6.2.8	stage queue	FINAL
	STORE_FORWARD && frameInStage	6.2.9		
	—	6.2.10	no frame selected	
FINAL	selected transfer completes	6.2.11	—	FIRST
	—	6.2.12	—	FINAL

~~**Row 6.2.6:** (Representing the store-and-forward option)~~
~~The primary transit queue is selected when a frame is available, to avoid queue overflows.~~
~~**Row 6.1.6:** In the absence of MAC-control frames, other transmission sources are checked.~~
~~**Row 6.2.7:** In the absence of MAC-control transmission credits, other transmission sources are checked.~~
~~**Row 6.1.8:** In the absence of class-A transmission credits, other transmission sources are checked.~~
~~**Row 6.1.9:** The MAC-control queue transmissions have precedence over client transmissions.~~
~~**Row** **6.2.8:** (Representing is the cut-through option)~~
The stage queue header pre-empts STQ transmissions, since provision checks were done previously.
Row 6.2.9: (Representing is the store-and-forward option)
A stage-queue frame pre-empts STQ transmissions, since provision checks were done previously.
Row 6.2.10: No frame is selected when no frame transmissions are possible.

Row 6.2.11: The next selection occurs when the transmission completes.
Row 6.2.12: The selected transmission/retransmission continues until completed.

Editors' Notes: *To be removed prior to final publication.*

These discussions of STQ are out of place. Need discussion of PTQ instead, matching the figure.

6.5 Dual-queue MAC design

The behavior of a dual-queue MAC is described by its transmit-selection protocols (described in this ~~subclause~~subclause) and shaping functions. The transmit-selection protocol and shaping functions are largely independent, but coupling (via the internal shaper provided *sendM* and *sendA* signals) is required to ensure conformance of MAC-supplied class-A control frame transmissions. Additional coupling (via the internal shaper provided *sendD* signal) is required to properly sustain downstream class-A transmissions.

A dual-queue MAC uses two transit queues, the primary transit queue (PTQ) for class-A traffic, and the secondary transit queue (STQ) for class-B and class-C traffic. The size of the secondary transit queue is left to the implementations. The dual-queue design is described in following subclauses.

6.5.1 Dual-queue MAC data paths

A dual-queue MAC makes use of two transit queues, as shown in Figure 6.13. The sizes of the transit queues are left to the implementations, although the primary transit queue has a minimum size of at least 2 MTUs. The size of the secondary transit queue determines its flow-control threshold values.

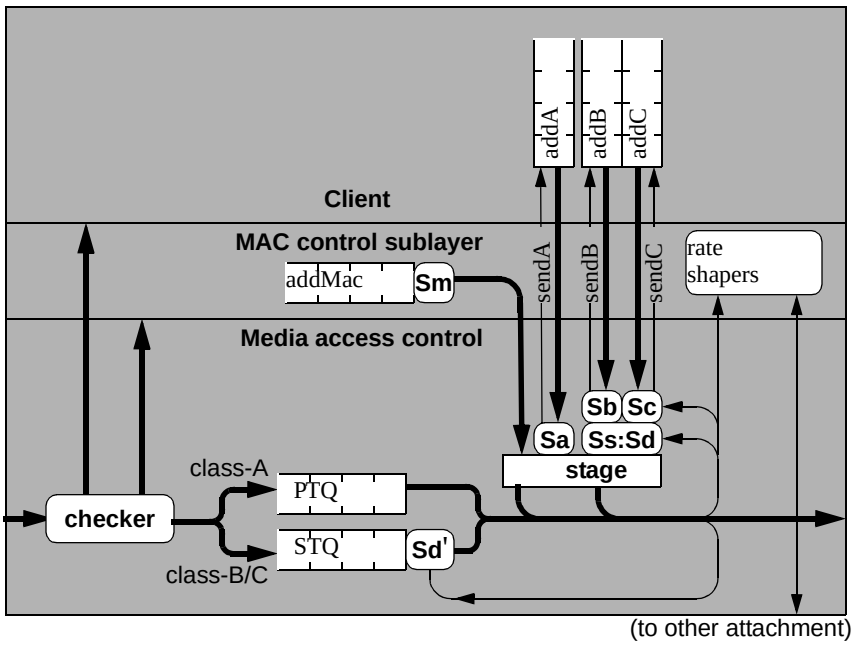

Figure 6.12—Dual-queue data-path model

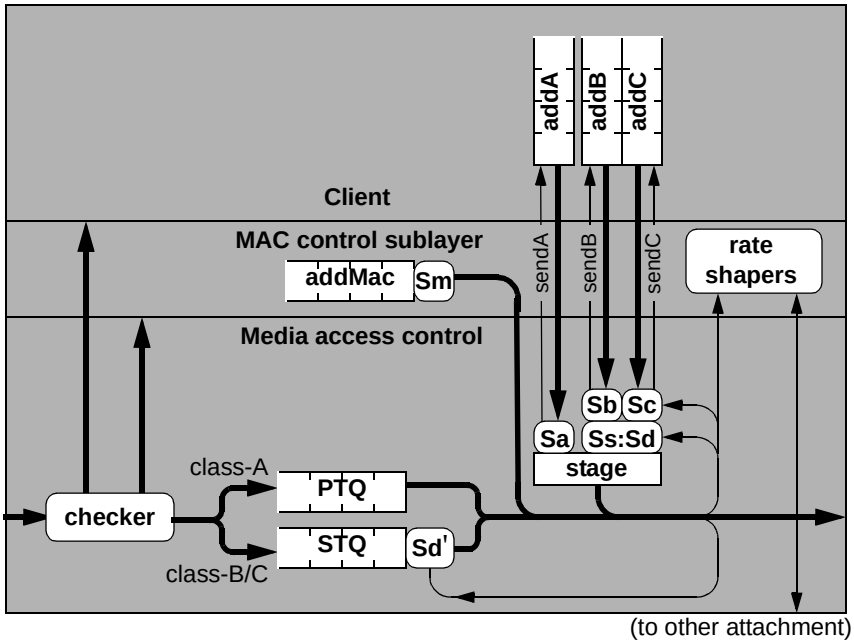

Figure 6.13—Dual-queue data-path model

Packets from the MAC-control send queue are rate-limited by shaper ***Sm*** (shaper of MAC). All class-A traffic is rate-limited by shaper ***Sa*** (shaper of class-A).

This standard does not define how to implement the PTQ and STQ queues. However, to meet ordering expectations, the following externally visible behaviors shall be supported:

a) PTQ ordering. A FIFO ordering shall be maintained when entries pass through the PTQ.
b) STQ ordering. A FIFO ordering shall be maintained when entries pass through the STQ.
c) Cross ordering. An entry from the STQ shall not be output before a previously received PTQ entry.

6.5.2 Dual-queue transmit selections

A dual-queue MAC can transmit data packets from three possible internal queues, as illustrated in Figure 6.15 An exact definition of the dual-queue transmit-frame selection sequence is specified in table 6.4, where the rows are evaluated in top-to-bottom order.

.

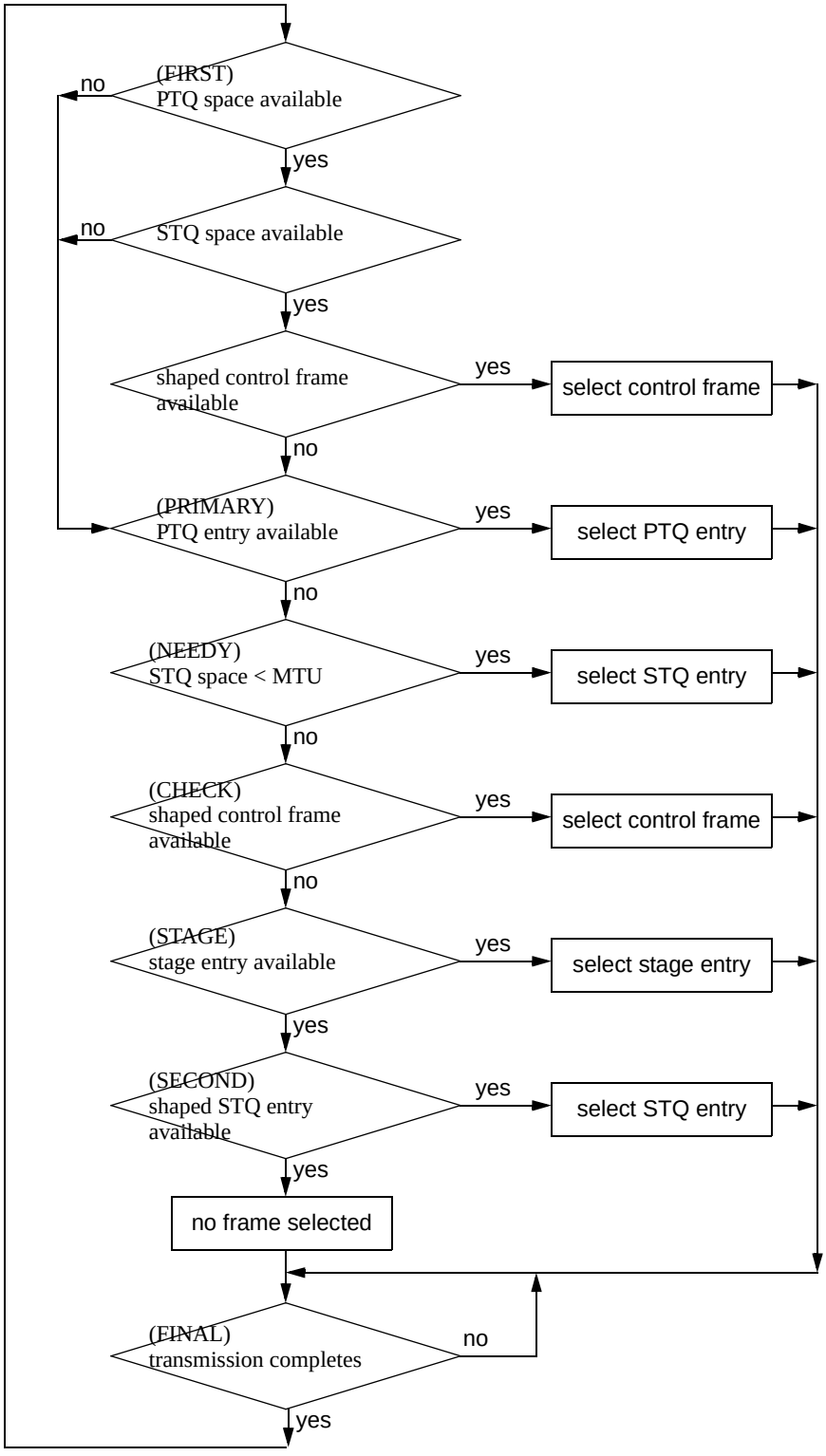

Figure 6.14—Dual-queue transmit selection

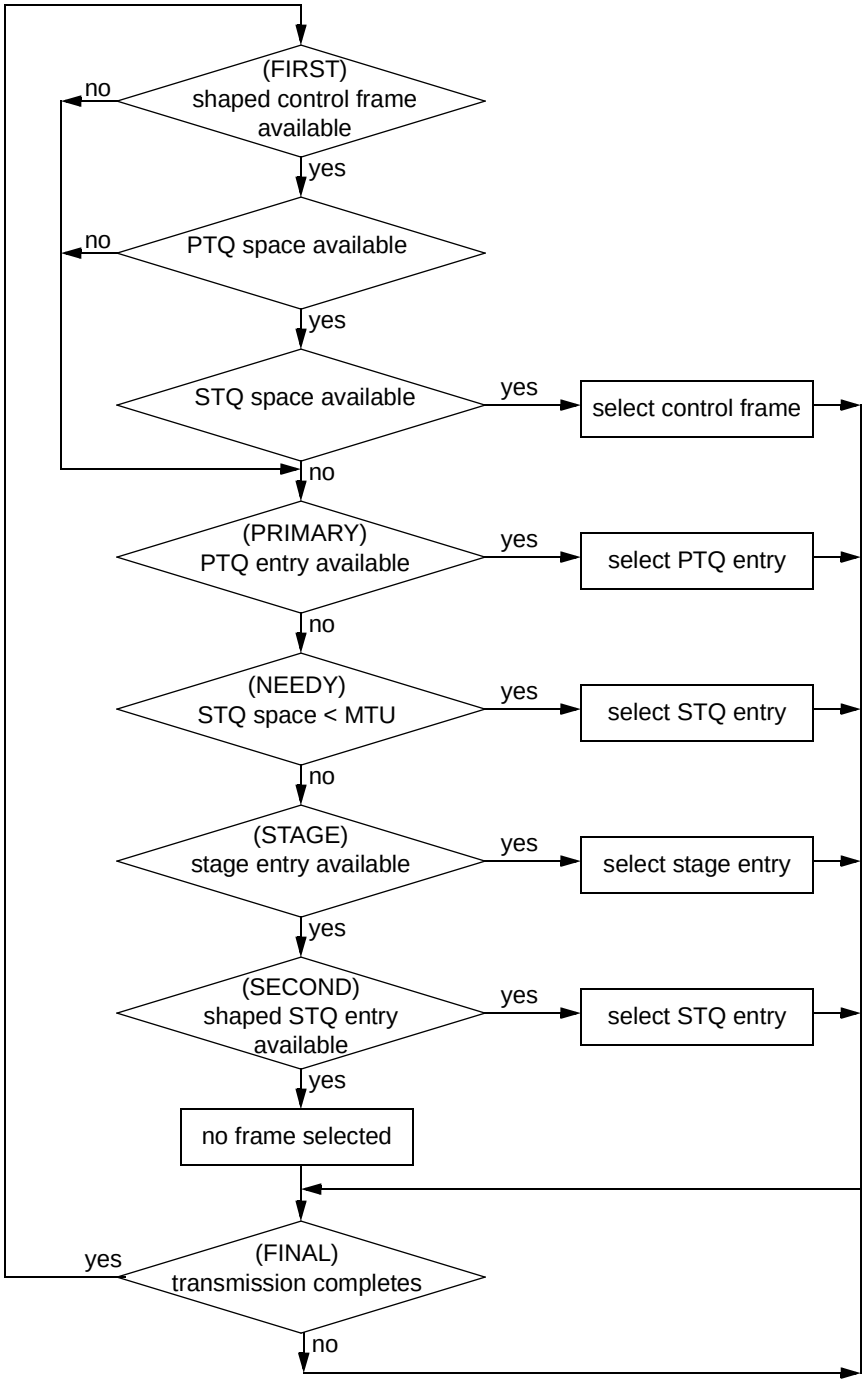

Figure 6.15—Dual-queue transmit selection

Acceptance into the stage entry is governed by the per class shapers and the fairness algorithm. Fairness ~~eligible~~ eligible traffic is added to the stage only when the station is not congested (as defined in clause 9).

Table 6.3—Dual-queue transmit-frame selection

Last state		Row	Next state	
state	**condition**		**selection**	**state**
FIRST	sizeOfMacControl > spaceInPTQ	6.3.1	—	PRIMARY
	sizeOfMacControl > spaceInSTQ	6.3.2		
	passM == 0	6.3.3		
	—	6.3.4	MAC control queue	FINAL
PRIMARY	CUT_THOUGH && headerInPTQ	6.3.5	primary transit queue	FINAL
	STORE_FORWARD && frameInPTQ	6.3.6		
	—	6.3.7	—	NEEDY
NEEDY	depthSTQ > sizeSTQ–MTU	6.3.8	secondary transit queue	FINAL
	—	6.3.9	—	CHECK
CHECK	NOT(entryInMacControl)	6.3.10	—	STAGE
	passM == 0	6.3.11	—	
	passA == 0	6.3.12	—	
	—	6.3.13	MAC control queue	FINAL
STAGE	CUT_THOUGH && headerInStage	6.3.14	stage buffer	FINAL
	STORE_FORWARD && frameInStage	6.3.15		
	—	6.3.16	—	SECOND
SECOND	passD==0	6.3.17	no transmit selection	FINAL
	CUT_THOUGH && headerInSTQ	6.3.18	secondary transit queue	
	STORE_FORWARD && frameInSTQ	6.3.19		
	—	6.3.20	no transmit selection	
FINAL	selected transfer completes	6.3.21	—	FIRST
	—	6.3.22	—	FINAL

~~**Row 6.4.2:** Validate that the MAC control frame is less than the available PTQ space.~~
~~**Row 6.4.3:** Validate that the MAC control frame is less than the available STQ space.~~

Table 6.4—Dual-queue transmit-frame selection

Last state		Row	Next state	
state	**condition**		**selection**	**state**
FIRST	frameInMacControl == 0	6.4.1	—	PRIMARY
	passM == 0	6.4.2		
	sizeOfMacControl > spaceInPTQ	6.4.3		
	sizeOfMacControl > spaceInSTQ	6.4.4		
	—	6.4.5	MAC control queue	FINAL
PRIMARY	CUT_THOUGH && headerInPTQ	6.4.6	primary transit queue	FINAL
	STORE_FORWARD && frameInPTQ	6.4.7		
	—	6.4.8	—	NEEDY
NEEDY	MTU > spaceInSTQ	6.4.9	secondary transit queue	FINAL
	—	6.4.10	—	STAGE
STAGE	CUT_THOUGH && headerInStage	6.4.11	stage buffer	FINAL
	STORE_FORWARD && frameInStage	6.4.12		
	—	6.4.13	—	SECOND
SECOND	passD==0	6.4.14	no transmit selection	FINAL
	CUT_THOUGH && headerInSTQ	6.4.15	secondary transit queue	
	STORE_FORWARD && frameInSTQ	6.4.16		
	—	6.4.17	no transmit selection	
FINAL	selected transfer completes	6.4.18	—	FIRST
	—	6.4.19	—	FINAL

~~**Row 6.4.4:** Validate that the MAC control frame remains within its provisioned rate.~~
~~**Row 6.4.5:** The small MAC-control transmissions have precedence over client transmissions.~~
~~**Row 6.4.6:** (Representing the cut-through option)~~
~~The primary transit queue is selected when a header is available, to avoid queue overflows.~~
~~**Row 6.4.7:** (Representing the store-and-forward option)~~
~~The primary transit queue is selected when a frame is available, to avoid queue overflows.~~
~~**Row 6.4.8:** In the absence of primary-transit-queue frames, other transmission sources are checked.~~
~~**Row 6.4.9:** The secondary transit queue is emptied when less than one MTU remains free, to avoid overflows.~~
~~**Row 6.4.10:** In the absence of secondary-transit-queue overflow threats, other transmission sources are checked.~~
~~**Row 6.3.10:** In the absence of MAC-control frames, other transmission sources are checked.~~
~~**Row 6.3.11:** In the absence of MAC-control transmission credits, other transmission sources are checked.~~
~~**Row 6.3.12:** In the absence of class-A transmission credits, other transmission sources are checked.~~
~~**Row 6.3.13:** The MAC-control queue transmissions have precedence over client transmissions.~~
~~**Row 6.4.11:** (Representing is the cut-through option)~~
~~The stage queue header pre-empts STQ transmissions, since provision checks were done previously.~~

~~**Row 6.4.12:** (Representing is the store-and-forward option)~~
~~A stage-queue frame pre-empts STQ transmissions, since provision checks were done previously.~~
~~**Row 6.4.13:** In the absence of stage-queue frames, other transmission sources are checked.~~
~~**Row 6.4.14:** When necessary to sustain downstream class-A traffic, STQ traffic is ignored.~~
~~**Row 6.4.15:** (Representing the cut-through option)~~
~~A secondary transit-queue header is serviced when available.~~
~~**Row 6.4.16:** (Representing the store-and-forward option)~~
~~A secondary transit-queue frame is serviced when available.~~
~~**Row 6.4.17:** The secondary transit queue is ignored when no frame is available.~~
~~**Row 6.4.18:** The next selection occurs after the transmission completes.~~
~~**Row 6.4.19:** The selected transmission/retransmission continues until completed.~~

Row 6.4.1: In the absence of MAC-control frames, other transmission sources are checked.
Row 6.4.2: In the absence of MAC-control transmission credits, other transmission sources are checked.
Row 6.4.3: In the absence of sufficient PTQ space, other transmission sources are checked.
Row 6.4.4: In the absence of sufficient STQ space, other transmission sources are checked.
Row 6.4.5: The MAC-control transmissions have precedence over client transmissions.

Row 6.4.6: (Representing the cut-through option)
The primary transit queue is selected when a header is available, to avoid queue overflows.
Row 6.4.7: (Representing the store-and-forward option)
The primary transit queue is selected when a frame is available, to avoid queue overflows.
Row 6.4.8: In the absence of primary-transit-queue frames, other transmission sources are checked.

Row 6.4.9: Empty a secondary transit queue when less than one MTU remains free (this avoid overflow).
Row 6.4.10: In the absence of a nearly full secondary-transit-queue, other transmission sources are checked.

Row 6.4.11: (Representing is the cut-through option)
The stage queue header pre-empts STQ transmissions, since provision checks were done previously.
Row 6.4.12: (Representing is the store-and-forward option)
A stage-queue frame pre-empts STQ transmissions, since provision checks were done previously.
Row 6.4.13: In the absence of stage-queue frames, other transmission sources are checked.

Row 6.4.14: When necessary to sustain downstream class-A traffic, STQ traffic is ignored.
Row 6.4.15: (Representing the cut-through option)
A secondary transit-queue header is serviced when available.
Row 6.4.16: (Representing the store-and-forward option)
A secondary transit-queue frame is serviced when available.
Row 6.4.17: The secondary transit queue is ignored when no frame is available.

Row 6.4.18: The next selection occurs after the transmission completes.
Row 6.4.19: The selected transmission/retransmission continues until completed.

6.6 Shaper details

6.6.1 Shaper-distinguished frames

For definitional simplicity, the shapers are described in terms of distinct algorithmically-similar shaping capabilities, as shown in Figure 6.16.

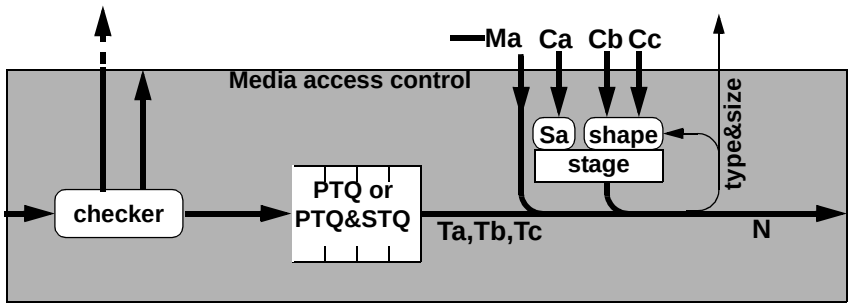

Figure 6.16—Shaper-distinguished frames

Each of the shapers is affected by the size and source of transmitted frames. For convenience, a shorthand notation is used to identify the sizes of distinct frame types, as listed below. Within these two-character names, the second characters of *a*, *b*, and *c* identify class-A, class-B, and class-C traffic respectively.

Ma	MAC-supplied frames
Ca	client supplied class-A frame
Cb	client supplied class-B frame
Cc	client supplied class-C frame
Ta	transit supplied class-A frame
Tb	transit supplied class-B frame
Tc	transit supplied class-C frame
N	not-supplied (nonexistent) frame

6.6.2 Flow-control sendA generation

This subclause describes the generation of a *sendA* indication, a flow-control indiction generated by the MAC for the purpose of flow-controlling client supplied class-A traffic. The generation of this *sendA* indication assumes the availability of a *passA* indication from the class-A shaper (see 6.7.4).

The scalar-policing class-A flow-control indication, *sendA*, has zero and nonzero values during congested and uncongested conditions, as illustrated in figure 6.17. An exact definition of the *sendA* generation is specified in table 6.5, where rows are evaluated in top-to-bottom order.

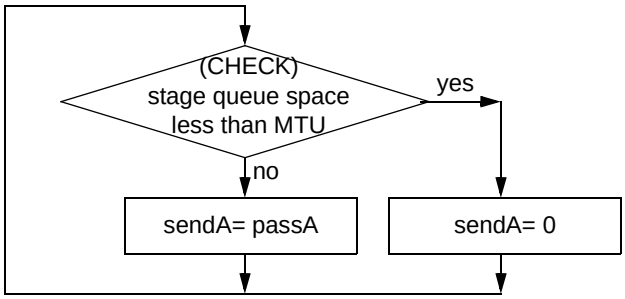

Figure 6.17—Flow-control sendA generation

Table 6.5—Flow-control sendA generation

Last state	Condition	Row	Action	Next state
CHECK	depthStageBuffer < (sizeStageBuffer–MTU)	6.5.1	sendA= 0	CHECK
	—	6.5.2	sendA= passA	

Note:
passA is the output of the class-A shaper (see Table 6.13)

Row 6.5.1: At least one MTU of stage-queue storage is required to safely buffer client-supplied frames.
Row 6.5.2: Setting *sendA* to the *passA* value enables throttled class-A transmissions when stage-buffer space is available. The class-A shaper provides the *passA* permission indication (see 6.7.4) based on the measured client-supplied transmission rate of class-A traffic.

6.6.3 Flow-control sendB generation

This subclause describes the generation of a *sendB* indication, a flow-control indiction generated by the MAC for the purpose of flow-controlling client supplied class-B traffic. For mono-queue designs, the generation of this *sendB* indication assumes the availability of *passB* and *passD* indications from class-B (see 6.7.5) and downstream (see 6.7.7) shapers respectively. For dual-queue designs, the generation of this *sendB* indication assumes the availability of *passB*, *passD*, and *passS* indications from class-B (see 6.7.5), downstream (see 6.7.7), and source (see 6.7.2) shapers respectively.

The scalar-policing of MAC-supplied class-B flows generates zero and nonzero *sendB* indications during congested and uncongested conditions, as illustrated in figure 6.18. An exact definition of the *sendB* generation is specified in table 6.6, where rows are evaluated in top-to-bottom order.

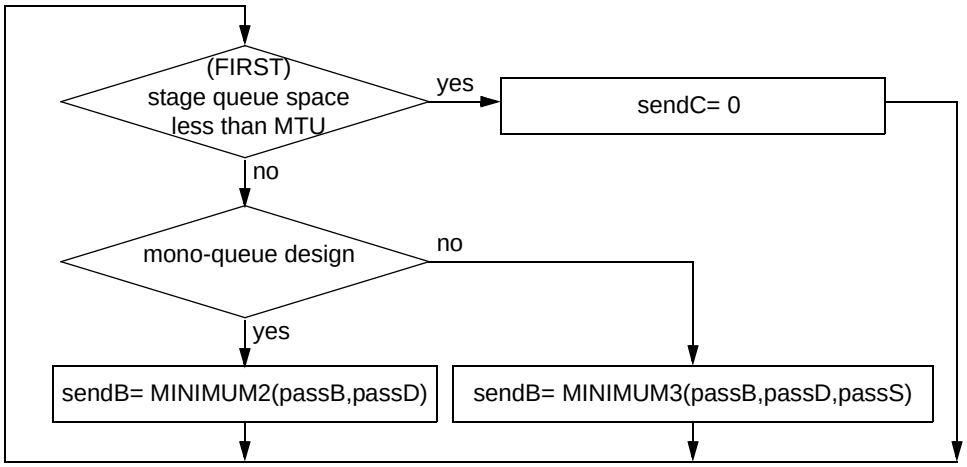

Figure 6.18—Flow-control sendB generation

Table 6.6—Flow-control sendB generation

Last state	Condition	Row	Action	Next state
CHECK	stage queue space less than MTU	6.6.1	sendB=0	FIRST
	monoQueueDesign	6.6.2	sendB= MINIMUM2(passB,passD)	
	—	6.6.3	leastB= MINIMUM3(passB,passD,passS)	

Note:
passB is the output of the class-B shaper (see Table 6.15)
passD is the output of the class-D shaper (see Table 6.17)

Row 6.6.1: At least one MTU of stage-queue storage is required to safely buffer client-supplied frames.
Row 6.6.2: Mono-queue design.
Setting *sendB* to the maximum of *passB* and *passD* values enables throttled class-B transmissions when stage-buffer space is available. One shaper provides the *passB* permission indication (see 6.7.4) based on the measured client-supplied transmission rate of class-B traffic; the other shaper provides the *passD* permission indication (see 6.7.7) based on the need to sustain downstream class-A traffic.
Row 6.6.3: Dual-queue design:
Setting *sendB* to the maximum of *passB*, *passD*, and *passS* values enables throttled class-B transmissions

when stage-buffer space is available. The final shaper provides a *passS* permission indication (see 6.7.2) based on the need to sustain transit-queue transmissions.

6.6.4 Flow-control sendC generation

This subclause describes the *sendC* indication, assuming the availability of *passC* and *passD* indications from weighted-fairness and downstream (see 6.7.7) shapers respectively; for dual-queue shapers a *passS* indications from the source shaper (see 6.7.2) is also assumed.

The shaping of MAC-supplied class-C flows generates zero and nonzero *sendC* indications during congested and uncongested conditions, as illustrated in figure 6.19. An exact definition of the *sendC* generation is specified in table 6.7, where rows are evaluated in top-to-bottom order.

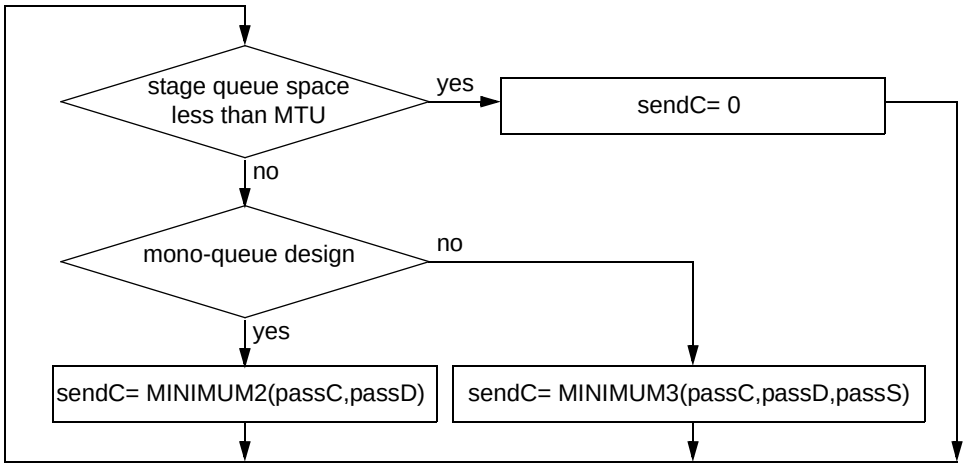

Figure 6.19—Flow-control sendC generation

Table 6.7—Flow-control sendC generation

Last state	Condition	Row	Action	Next state
CHECK	spaceInStage < MTU	6.7.1	sendC=0	CHECK
	monoQueueDesign	6.7.2	sendC= MINIMUM2(passC,passD)	
	—	6.7.3	sendC= MINIMUM3(passC,passD,passS)	

Note:
passD is the output of the class-D shaper (see Table 6.17)
passS is the output of the source shaper (see Table 6.9)

Row 6.7.1: At least one MTU of stage-queue storage is required to safely buffer client-supplied frames.
Row 6.7.2: Mono-queue design.
Setting *sendC* to the maximum of *passC* and *passD* values enables throttled class-C transmissions when stage-buffer space is available. One shaper provides the *passC* permission indication (see 6.7.4) based on the fairness rate of class-C traffic; the other shaper provides the *passD* permission indication (see 6.7.7) based on the need to sustain downstream class-A transmissions.
Row 6.7.3: Dual-queue design.
Setting *sendC* to the maximum of *passC*, *passD*, and *passS* values enables throttled class-B transmissions

when stage-buffer space is available. The final shaper provides a *passS* permission indication (see 6.7.2) based on the need to sustain transit-queue transmissions.

6.7 MAC shapers

The shapers described in the following subclauses generate the *passA*, *passB*, *passC*, *passD*, and *passC* values used to generate the *sendA*(see 6.6.2), *sendB*(see 6.6.3), and *sendC*(see 6.6.4) indications that are supplied to the client.

6.7.1 Shaper summary

Each MAC rate shaper can be readily identified by its credit-value name, as listed in Table 6.8. Some of the transmission paths are affected by only one of these shapers; others are influenced by a cascaded string of shapers.

Table 6.8—Rate-shaper summary

Description	Row	credits	shaper	References
Source provisions, to sustain transit transmissions	6.8.1	creditS	Ss	6.7.2
MAC-control provisioned transmissions	6.8.2	creditM	Sm	6.7.3
Class-A provisioned transmissions	6.8.3	creditA	Sa	6.7.4
Class-B provisioned transmissions	6.8.4	creditB	Sb	6.7.5
Sustain downstream class-A transmissions	6.8.5	creditD	Sd	6.7.7

6.7.2 Source shaper

6.7.2.1 Source shaper parameter

The client shaper is used to partition available downstream bandwidths between client and STQ transmissions. To perform this function, the shaper credits are decremented on client transmissions and incremented on STQ transmissions, as illustrated in Figure 6.20. Positive credits are cleared when no stage entry is available and debits (negative credits) are cleared when no STQ entry is available.

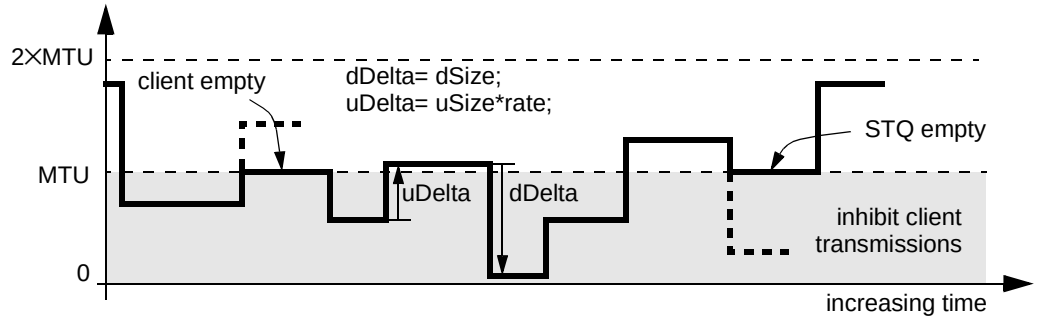

Figure 6.20—Source shaper adjustments

NOTE—A simple instance of this shaper has a rate of 1/2 or 1, as described in Figure 6.7.2.2. For such instances, the 1/2 rate involves decrementing and incrementing credits by the sizes of transmitted client-supplied class-B/C and STQ-supplied class-B/C frames respectively. When the rate change to 1, these increments are simply skipped.

NOTE—This shaper is a special case of the shaper described in 6.6, but has been described separately to clarify the distinctive rate-balancing properties.

6.7.2.2 Source shaper rates

The *rateS* (rate of secondary) secondary-rate limiting parameter depends on the secondary-transit-queue (STQ) depth, as shown in Figure 6.21. Any monotonically increasing functions within the shaded polygon of Figure 6.21 is allowed, although one of the two dotted-line functions should be implemented. One strategy is to limit the cumulative send-B and send-C transmissions, to bound the worst-case STQ delay. Another strategy is to back off the send-B and send-C transmissions when the secondary transit queue reaches a LO_THRESHOLD condition, so that sufficient STQ space is left for conflicting class-A transmissions.

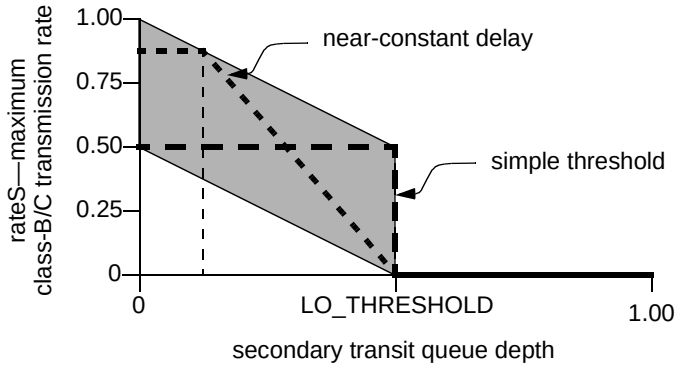

Figure 6.21—Depth-dependent source-shaper *rateS* parameter

6.7.2.3 Source shaper parameters

The client shaper adjusts the ***creditS*** value to balance class-B/C client-supplied transmissions and transit-supplied transmissions, as listed in table 6.9. To encourage client-supplied transmissions without starving transit-supplied STQ transmissions, the rate value ***rateS*** depends on the STQ depth (see 6.7.2). The Cb/Cc and Tb/Tc values represent sizes of transit-sourced class-B/C traffic and client-sourced class-B/C traffic respectively.

Table 6.9—Transit-shaper parameters

Values		Amounts		hiLimit		loLimit		output
credits	**rate**	**dSize**	**uSize**	**Name**	**Value**	**Name**	**Value**	**testS**
creditS	rateS	Cb,Cc	Tb,Tc	hiLimitS	MTU	loLimitS	MTU	creditS>loLimitS

#define passS (testS ? MAX_STATIONS : 0)

The credits decrease by the rate-scaled size of transit-supplied class-B and class-C frames; they increase by the rate-scaled size of client-supplied class-B/class-C frames.

6.7.2.4 Source shaper resets

The source shaper credits are adjusted to avoid credit accumulation during inactive periods, as listed in Figure 6.17. An exact definition of the restored retransmit credits sequence is specified in table 6.10, where the rows are evaluated in top-to-bottom order.

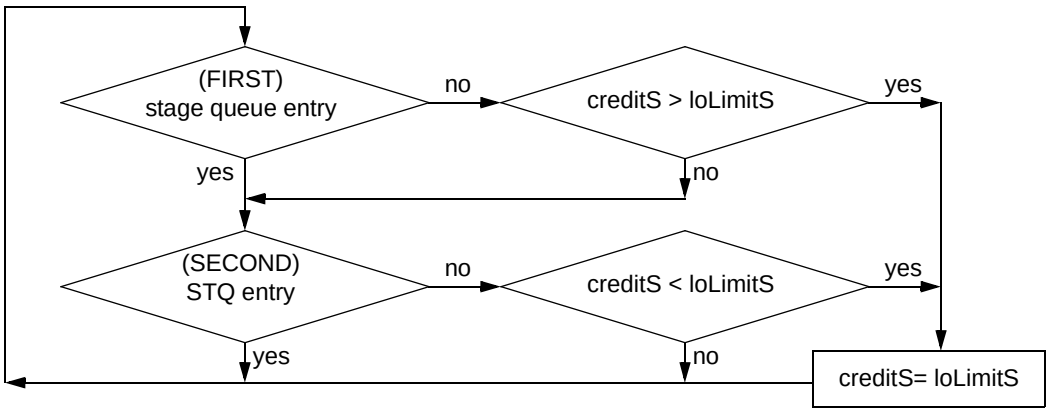

Figure 6.17—Source shaper resets

Table 6.10—Source shaper resets

Last state		Row	Next state	
state	**condition**		**action**	**state**
FIRST	depthStageQueue > 0	6.10.1	—	FINAL
	creditS < loLimitS	6.10.2	creditS= loLimitS	
	—	6.10.3	—	
FINAL	depthSTQ > 0	6.10.4	—	FIRST
	creditS > loLimitS	6.10.5	creditS= loLimitS	
	—	6.10.6	—	

Row 6.10.1: Credits aren't cancelled while frames remain queued.
Row 6.10.2: Credits are cancelled when transmission opportunities are passed.
Row 6.10.3: Checks are also made for debit cancellations.

Row 6.10.4: Debits aren't cancelled while frames remain queued.
Row 6.10.5: Debits are cancelled when retransmission opportunities are passed.
Row 6.10.6: Checks are also made for credit cancellations.

6.7.3 Control shaper

6.7.3.1 Control shaper parameters

With control shaper limits the MAC-supplied class-A traffic to its provisioned limits, based on the parameters listed in table 6.11

Table 6.11—Control shaper parameters

Values		Amounts		hiLimit		loLimit		output
credits	**rate**	**dSize**	**uSize**	**Name**	**Value**	**Name**	**Value**	**testM**
creditM	rateM	Ma	–all–	hilimitM	MTU	lolimitM	MTU	creditM>lolimitM

#define passM (testM ? MAX_STATIONS : 0)

As indicated by the dSize column, the *creditM* credits decrease by the scaled size of a MAC-supplied class-A frame transmissions. As indicated by the uSize column, these credits increase by the rates sizes of all other (client-supplied class-A, client-supplied class-B, client-supplied class-C, transit-supplied class-A, transit-supplied class-B, transit-supplied class-C, and null) frame transmissions.

6.7.3.2 Control shaper resets

The control-frame credits are adjusted to avoid credit accumulation during inactive periods, as listed in figure 6.22. An exact definition of these credit-restoring conditions is specified in table 6.12.

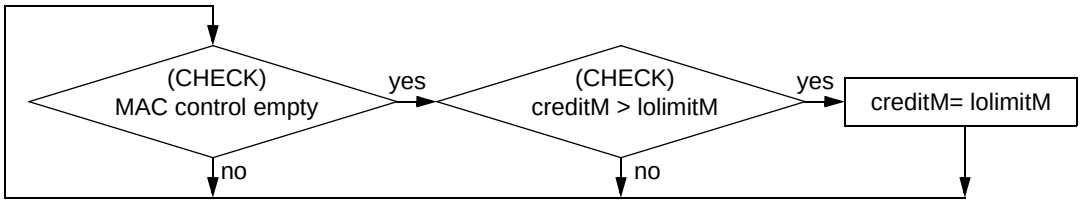

Figure 6.22—Control shaper resets

Table 6.12—Control shaper resets

Last state		Row	Next state	
state	**condition**		**action**	**state**
FIRST	depthMacControl > 0	6.12.1	—	FIRST
	creditM > lolimitM	6.12.2	creditM= lolimitM	
	—	6.12.3	—	

Row 6.12.1: Accumulated credits don't disappear unless the MAC control queue has emptied.
Row 6.12.2: Accumulated credits disappear when the MAC control queue has emptied.
Row 6.12.3: Accumulated debits don't disappear when the MAC control queue has emptied.

6.7.4 Class-A shaper

6.7.4.1 Class-A shaper parameters

The class-A shaper limits the class-A transmissions to their provisioned limits, using the parameters listed in table 6.13.

Table 6.13—Class-A shaper parameters

Values		Amounts		hiLimit		loLimit		output
credits	**rate**	**dSize**	**uSize**	**Name**	**Value**	**Name**	**Value**	**testA**
creditA	rateA	Ca,Ma	–all–	hiLimitA	limitA	loLimitA	MTU	creditA>loLimitA

#define limitA TBD
#define passA (testA ? MAX_STATIONS : 0)

As indicated by the dSize column, the *creditA* credits decrease by the scaled sizes of a client-supplied class-A and MAC-supplied class-A frame transmissions. As indicated by the uSize column, these credits increase by the scaled sizes of transit-supplied class-A, transit-supplied class-B, transit-supplied class-C, client-supplied class-B, client-supplied class-C, and null frames.

6.7.4.2 Class-A shaper resets

The class-A shaper credits are adjusted to avoid credit accumulation during inactive periods, as listed in figure 6.23. An exact definition of these credit-reset conditions is specified in table 6.14, where rows are evaluated in top-to-bottom order.

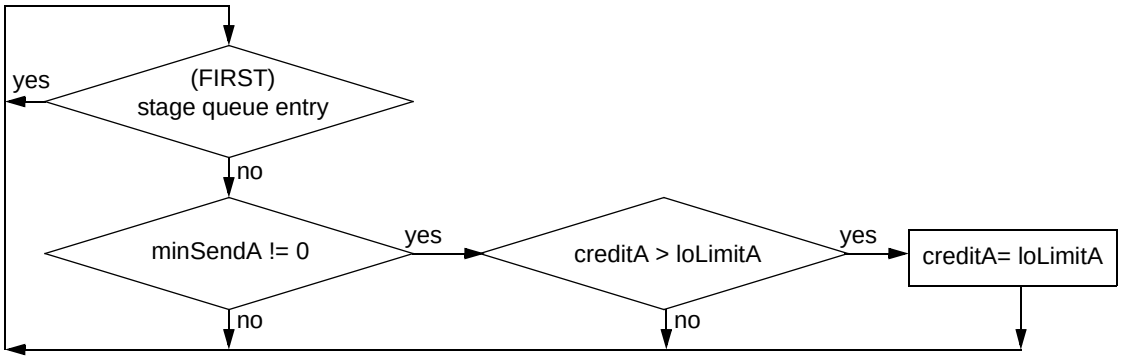

Figure 6.23—Class-A shaper resets

Table 6.14—Class-A shaper resets

Last state		Row	Next state	
state	**condition**		**action**	**state**
FIRST	depthStageBuffer > 0	6.14.1	—	FIRST
	minSendA == 0	6.14.2		
	creditA <= loLimitA	6.14.3		
	—	6.14.4	creditA= loLimitA	

Row 6.14.1: The stage buffer hasn't emptied, which implies some traffic was ready to be sent.
Row 6.14.2: The client was able to send class-A for more than a full handshake interval.

Row 6.14.3: The debits (negative credits) are never reset, as that could allow unshaped transmissions.
Row 6.14.4: Positive credits are discarded when there is nothing to send.

6.7.5 Class-B shaper

6.7.5.1 Class-B shaper parameters

The class-B shaper limits class-B frame transmissions to their provisioned levels, based on the parameters listed in table 6.15.

Table 6.15—Client class-B shaper parameters

Values		Amounts		hiLimit		loLimit		output
credits	**rate**	**dSize**	**uSize**	**Name**	**Value**	**Name**	**Value**	**testB**
creditB	rateB	Cb	–all–	hiLimitB	limitB	loLimitB	MTU	creditB>loLimitB

#define limitB TBD
#define passB (testB ? MAX_STATIONS : 0)

As indicated by the dSize column, the *creditB* credits decrease by the scaled sizes of client-supplied class-B frames. As indicated by the uSize column, they increase by the scaled sizes of all other (client-supplied class-B, client-supplied class-C, MAC-supplied class-A, transit-supplied class-A, transit-supplied class-B, transit-supplied class-C, and null) frames.

6.7.5.2 Class-B shaper resets

The class-B shaper credits are adjusted to avoid credit accumulation during inactive periods, as listed in figure 6.24. An exact definition of these credit-restoring conditions is specified in table 6.16, where rows are evaluated in top-to-bottom order.

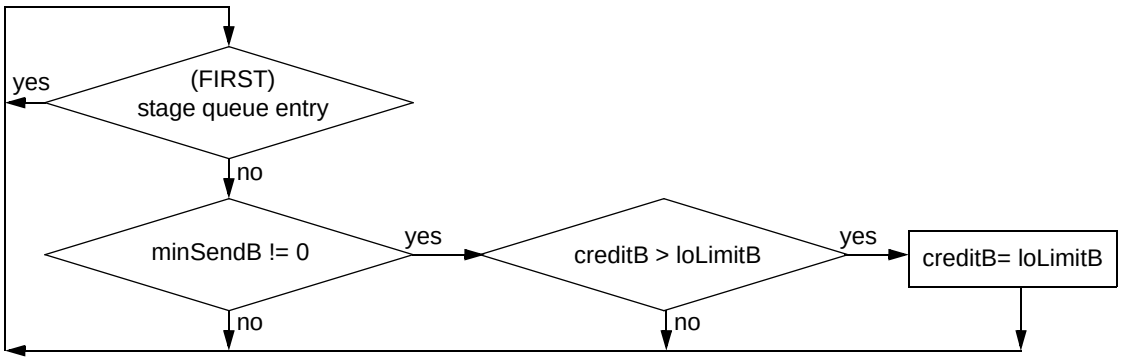

Figure 6.24—Class-B shaper resets

Table 6.16—Class-B shaper resets

Last state		Row	Next state	
state	**condition**		**action**	**state**
FIRST	depthStageBuffer > 0	6.16.1	—	FIRST
	minSendB == 0	6.16.2		
	creditB <= loLimitB	6.16.3		
	—	6.16.4	creditB= loLimitB	

Row 6.16.1: The stage buffer hasn't emptied, which implies some traffic was ready to be sent.
Row 6.16.2: The client was able to send class-B for more than a full handshake interval.

Row 6.16.3: The debits (negative credits) are never reset, as that could allow unshaped transmissions.
Row 6.16.4: Positive credits are discarded when there is nothing to send.

6.7.6 Class-C shaper

See the fairness clause.

6.7.7 Downstream shaper

Editors' Notes (DVJ): To be removed prior to final publication.

Further analysis is needed to consolidate the requirements for monitoring class-A0 as well as class-A1.

6.7.7.1 Downstream shaper parameters

The downstream shaper monitors the transit traffic to ensure sufficient levels if sustainable class-A traffic for downstream stations, as listed in table 6.17.

Table 6.17—Downstream shaper parameters

Values		Amounts		hiLimit		loLimit		output
credits	**rate**	**dSize**	**uSize**	**Name**	**Value**	**Name**	**Value**	**testD**
creditD	rateD	Ca,Ma, Ta,N	–all–	hiLimitD	limitA	loLimitD	MTU	creditD>loLimitD

// See tablexx for limitA definition
#define rateD (backlog ? maxA:maxA0)
#define passD (testD ? MAX_STATIONS : 0)

As the *rateD* value (within the rate column) depends on the *backlog* congestion indication received from downstream stations. If the backlog value is 1, then the *maxA* value is used to sustain the worst-case downstream class-A traffic. Otherwise, the (never larger, sometimes smaller) *maxA0* value is used to sustain the minimum level of downstream class-A traffic. The values of *maxA* and *maxA0* represent the cumulative class-A and subclass-A0 traffic, as provisioned over any of the configured links (see xx).

As indicated by the dSize column, the *creditD* decrease by the scaled sizes of client-supplied class-A, MAC-supplied class-A, transit-supplied class-A, and nonexistent frames. As indicated by the dSize column, they increase by the scaled sizes of transit-supplied class-B, transit-supplied class-C, client-supplied class-B, and client-supplied class-C frames.

6.8 Receive operation

The receive operation is shown in Figure 6.20. A corrupted frame is immediately rejected, a remaining accept frame is copied to the client or MAC control, the *timeToLive* field is decremented, and a passing-through frame is saved into the primary or secondary transmit queue. The mono-queue and dual-queue operations are both defined; the queue-design dependent portions are shaded in grey.

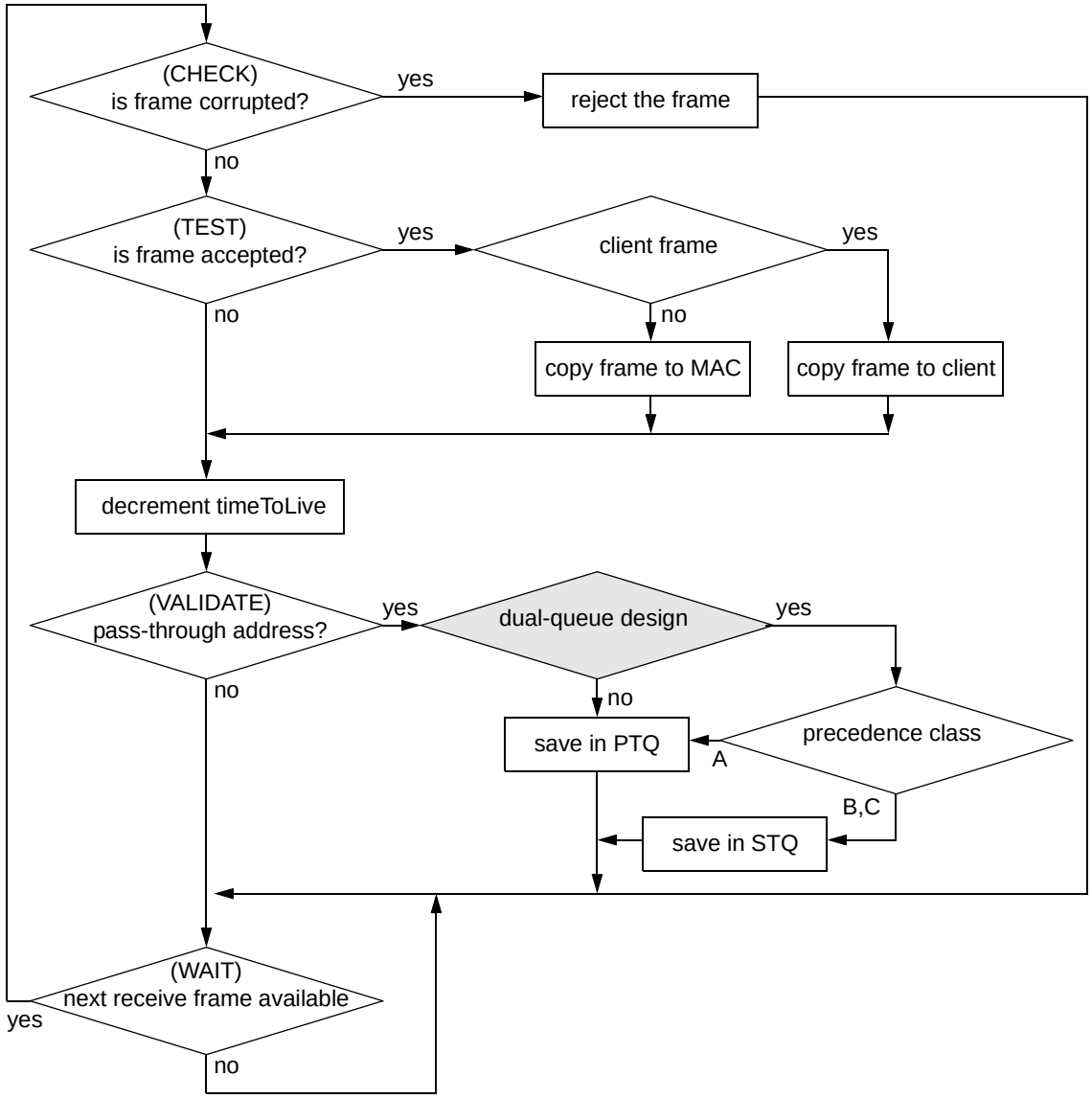

Figure 6.20—Receive processing flowchart

The specific frame processing steps are specified in Table 6.19. The mono-queue and dual-queue operations are both defined; the queue-design dependent portions are shaded in grey.

Table 6.18—Receive processing states

Last state		Row	Next state	
state	condition		action	state
CHECK1	headerCRC == CRC(header)	61	—	CHECK2
	—	62	—	CHECK1
CHECK2	dataCRC==CRC(data)	63	—	CHECK3
	dataCRC== ~CRC(data)	64	—	
	—	65	dataCRC= ~CRC(data) errorCount+= 1	
CHECK3	timeToLive==0	66	—	START
	MULTICAST(sourceMacAddress)	67		
	header.wrap==0&&header.ringID != myRingID	68		
	—	69	—	TEST
TEST	mode.permiscous==1	60	—	COPY
	header.ringID != myRingID	61	—	SENDING
	destinationMacAddress==myMacAddress	62	—	COPY
	sourceMacAddress==myMacAddress	63	—	SENDING
	MULTICAST(destinationMacAddress)	64	—	COPY
	frameHeader.flooding==1	65		
	—	66	—	SENDING
COPY	DataFrame(frameHeader)	67	CopyToClient()	SENDING
	—	68	CopyToMac()	
SENDING	—	69	timeToLive–= 1	VALIDATE

Table 6.18—Receive processing states

Last state		Row	Next state	
state	condition		action	state
VALIDATE	timeToLive==0	60	—	START
	header.wrap==0&&myState.wrapped	61		
	header.ringID != myState.ringID	62	RecomputeHeaderCrc()	SAVE
	destinationMacAddress=myMacAddress	63	—	START
	sourceMacAddress=myMacAddress	64		
	mono transit-queue implementation	65	CopyToPTQ()	START
	frameHeader.class==CLASS_A	66		
	—	67	CopyToSTQ()	
WAIT	next receive frame available	68	—	CHECK1
	—	69	—	WAIT

Table 6.19—Receive processing states

Last state		Row	Next state	
state	condition		action	state
CHECK1	headerCRC == CRC(header)	6.19.1	—	CHECK2
	—	6.19.2	—	CHECK1
CHECK2	dataCRC==CRC(data)	6.19.3	—	CHECK3
	dataCRC== ~CRC(data)	6.19.4	—	
	—	6.19.5	dataCRC= ~CRC(data) errorCount+= 1	
CHECK3	timeToLive==0	6.19.6	—	START
	MULTICAST(sourceMacAddress)	6.19.7		
	header.wrap==0&& header.ringID != myRingID	6.19.8		
	—	6.19.9	—	TEST

Table 6.19—Receive processing states

Last state		Row	Next state	
state	**condition**		**action**	**state**
TEST	mode.permiscous==1	6.19.10	—	COPY
	header.ringID != myRingID	6.19.11	—	SENDING
	destinationMacAddress==myMacAddress	6.19.12	—	COPY
	sourceMacAddress==myMacAddress	6.19.13	—	SENDING
	MULTICAST(destinationMacAddress)	6.19.14	—	COPY
	frameHeader.flooding==1	6.19.15		
	—	6.19.16	—	SENDING
COPY	DataFrame(frameHeader)	6.19.17	CopyToClient()	SENDING
	—	6.19.18	CopyToMac()	
SENDING	—	6.19.19	timeToLive–= 1	VALIDATE
VALIDATE	timeToLive==0	6.19.20	—	START
	header.wrap==0&&myState.wrapped	6.19.21		
	header.ringID != myState.ringID	6.19.22	RecomputeHeaderCrc()	SAVE
	destinationMacAddress=myMacAddress	6.19.23	—	START
	sourceMacAddress=myMacAddress	6.19.24		
	mono transit-queue implementation	6.19.25	CopyToPTQ()	START
	frameHeader.class==CLASS_A	6.19.26		
	—	6.19.27	CopyToSTQ()	
WAIT	next receive frame available	6.19.28	—	CHECK1
	—	6.19.29	—	WAIT

~~**Row 6.19.1:** If the header CRC differs from its expected value, the potentially corrupt frame is rejected.~~
~~**Row 6.19.2:** Otherwise, frame processing continues in the normal fashion.~~
~~**Row 6.19.3:** A good data CRC is processed normally.~~
~~**Row 6.19.4:** A stomped data CRC is flagged and processed normally.~~
~~**Row 6.19.5:** A stomped data CRC is counted, flagged, and processed normally.~~
~~**Row 6.19.6:** Expired frames are immediately rejected.~~
~~**Row 6.19.7:** Multicast source addresses are illegal and therefore rejected.~~
~~**Row 6.19.8:** Steer-only frames are rejected when apparently wrapped.~~
~~**Row 6.19.9:** Otherwise, normal frame processing continues.~~
~~**Row 6.19.11:** Wrapped frames are not copied when returning on their opposing run.~~
~~**Row 6.19.14:** Multicast frames are accepted by all but their final destination station.~~
~~**Row 6.19.12:** Unicast frames are accepted at their destination station.~~
~~**Row 6.19.15:** Flooded frames are accepted by intermediate stations.~~
~~**Row 6.19.16:** Otherwise, frames are not accepted.~~
~~**Row 6.19.17:** Accepted data frames are passed to the client.~~
~~**Row 6.19.18:** Otherwise, accepted frames are passed to the control portion of the MAC.~~
~~**Row 6.19.19:** The *timeToLive* field is always updated on passing-through frames.~~
~~**Row 6.19.20:** Expired frames are rejected.~~
~~**Row 6.19.21:** Steer-only frames are rejected when the wrapping station is reached.~~
~~**Row 6.19.22:** Wrapped frames are not stripped when returning on their opposing run.~~
~~**Row 6.19.23:** Frames are rejected after reaching their destination.~~
~~**Row 6.19.24:** Frames are rejected after returning to their source.~~
~~**Row 6.19.25:** (This row is only applicable to the dual-queue design option.)~~
~~In mono-queue designs, all remaining frames are placed into the primary transit queue.~~
~~**Row 6.19.26:** The remaining class-A frames are placed into the primary transit queue.~~
~~**Row 6.19.27:** The remaining class-B and class-C frames are placed into the secondary transit queue.~~

Editors' Notes: *To be removed prior to final publication.*

These row values need to be checked to make sure they align correctly with the figure.

Row 6.19.1: If the header CRC differs from its expected value, the potentially corrupt frame is rejected.
Row 6.19.2: Otherwise, frame processing continues in the normal fashion.

Row 6.19.3: A good data CRC is processed normally.
Row 6.19.4: A stomped data CRC is flagged and processed normally.
Row 6.19.5: A stomped data CRC is counted, flagged, and processed normally.

Row 6.19.6: Expired frames are immediately rejected.
Row 6.19.7: Multicast source addresses are illegal and therefore rejected.
Row 6.19.8: Steer-only frames are rejected when apparently wrapped.
Row 6.19.9: Otherwise, normal frame processing continues.

Row 6.19.11: Wrapped frames are not copied when returning on their opposing run.
Row 6.19.14: Multicast frames are accepted by all but their final destination station.
Row 6.19.12: Unicast frames are accepted at their destination station.
Row 6.19.15: Flooded frames are accepted by intermediate stations.
Row 6.19.16: Otherwise, frames are not accepted.

Row 6.19.17: Accepted data frames are passed to the client.
Row 6.19.18: Otherwise, accepted frames are passed to the control portion of the MAC.

Row 6.19.19: The *timeToLive* field is always updated on passing-through frames.

Row 6.19.20: Expired frames are rejected.
Row 6.19.21: Steer-only frames are rejected when the wrapping station is reached.
Row 6.19.22: Wrapped frames are not stripped when returning on their opposing run.
Row 6.19.23: Frames are rejected after reaching their destination.
Row 6.19.24: Frames are rejected after returning to their source.

Row 6.19.25: (This row is only applicable to the dual-queue design option.)
In mono-queue designs, all remaining frames are placed into the primary transit queue.
Row 6.19.26: The remaining class-A frames are placed into the primary transit queue.
Row 6.19.27: The remaining class-B and class-C frames are placed into the secondary transit queue.

6.9 Wrappable data paths

There are 2 methods of implementing wrapping, each with different effects.

Editors' Notes: *To be removed prior to final publication.*

Which of the wrapping methods is standardized, or if both are standardized and either is optional is still under discussion. The implications on the client need to be described.

6.9.1 Center wrap

Each wrapping capable station has wrappable data paths that allow frames to loop-back to the opposing ringlet after link failures are detected, as shown in Figure 6.26. The wrapping mode has no effect on the behavior of the attachment points, but steering-only frames are rejected when passing into a wrong-run attachment point. This will allow a station to be connected to the ring when there is a single attachment failure.

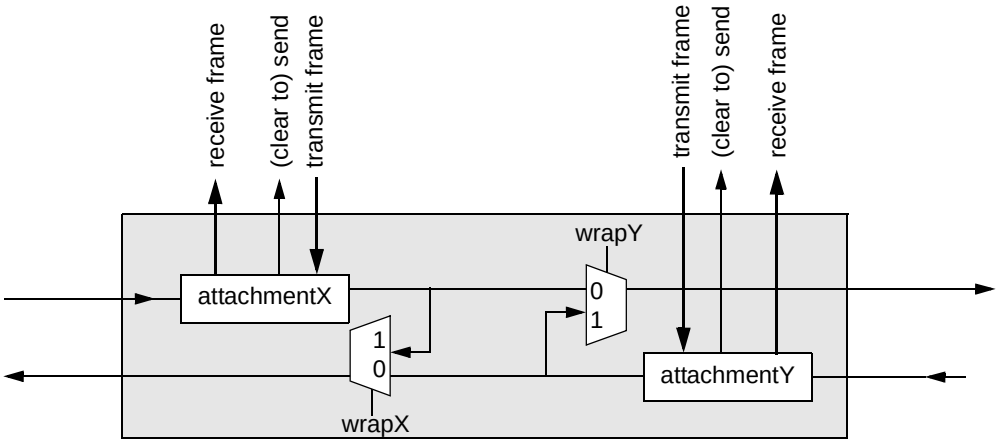

Figure 6.25—Wrappable data paths

6.9.2 Edge wrap

Each wrapping capable station has wrappable data paths that allow frames to loop-back to the opposing ringlet after link failures are detected, as shown in Figure 6.26. The wrapping mode has no effect on the behavior of the attachment points, but steering-only frames are rejected when passing into a wrong-run attachment point.

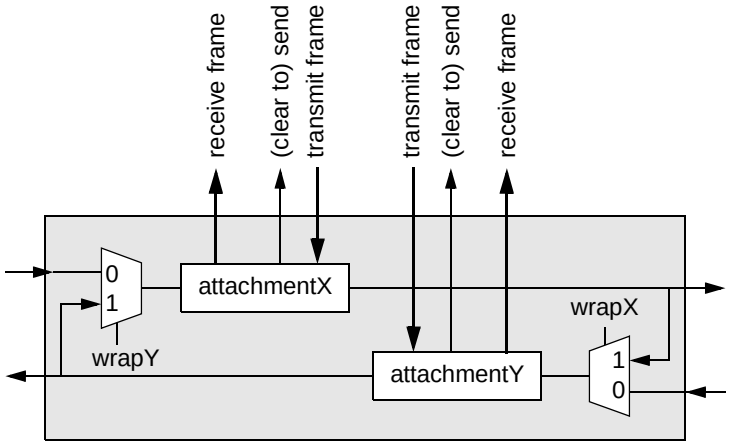

Figure 6.26—Wrappable data paths

Editors' Notes: *To be removed prior to final publication.* *Whether the wrapped traffic goes through the opposite attachment or bypasses it is under discussion.*