Proposed Text for Clauses 6, 11, 12, and Annexes J, K

Introduction

This document contains proposed text for Clauses 6, 11, and 12, and for Annexes J, K. Together they define the media access control (MAC) fairness algorithm used for RPR.

Comments on this proposal can be directed to the contributing editors:

Anoop Ghanwani
Lantern Communications
211 River Oaks Parkway
San Jose, CA 95134
USA
Phone: +1.408.521.6707
Fax: +1.408.521.6899
Email: anoop@lanterncom.com

Acknowledgments

Much of the text presented in this document has been taken from various proposals submitted in the past to the IEEE 802.17 Working Group. The editor would like to acknowledge the individuals that contributed to the text in those proposals. They include:

Adisak Mekkittikul, Alan Pepper, Albert Herrera, Angela Faber, Anoop Ghanwani, C.P. Fu, Harry Peng, Henry Hsiaw, Italo Busi, Jason Fan, Jeanne De Jaegher, John Hawkins, John Lemon, Kanaiya Vasani, Mannix O'Connor, Mark Holness, Nader Vijeh, Peter Lassen, Raj Sharma, Rhett Brikovskis, Robin Olssen, Sanjay Agrawal, Vittorio Mascolo, Wai-Chau Hui, Yiming Yao.

TABLE OF CONTENTS

6. MAC Data Path

6.1 Transit path requirements

The requirements of the MAC transit path are as follows.

1. The transit path is part of the shared medium.
2. The transit path is lossless.
3. The transit path implements destination address based and source address based stripping.
4. The transit path implements broadcasting and multicasting: receive and copy mode.
5. The transit path implements minimal buffering in order to:
   - Minimize the cost of the standard RPR MAC chip saving memory cost.
   - Minimize delay in the transit path.
   - Maximize scalability to allow RPR MAC chips to scale to higher speeds.

   The transit buffer is used only for collision avoidance.

Figure 6-1 shows the transit path design. The transit path consists of the header processing entity, the transit buffer and the arbiter entity. The transit path is considered an extension of the media, and hence is part of the ring.

The RPR MAC frame header-processing block determines reception and transit conditions. The role of the transit buffer is exclusively to avoid collisions between Add and Pass frames. The Add/Pass Scheduler arbitrates access to the outgoing link between Add and Pass frames.

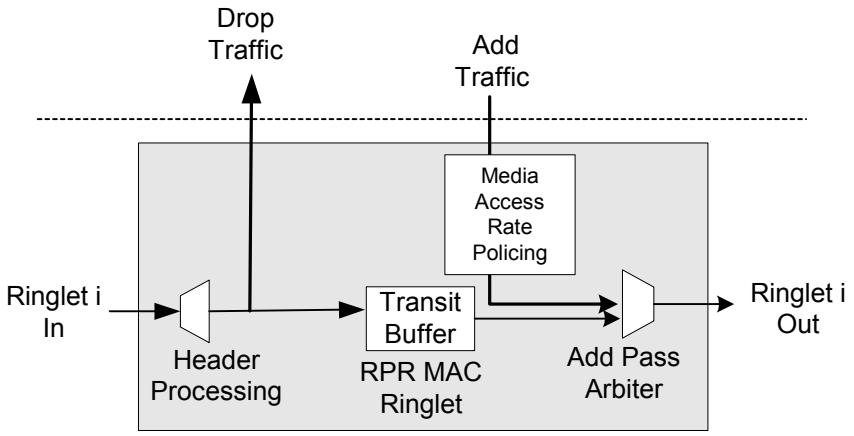

Figure 6-1: Transit path design

6.2 Reception rules

When a frame arrives on an ingress port of an RPR MAC, the destination MAC address is matched with the RPR database in the header-processing block shown in Figure 6-1. The decision of whether to receive and copy the frame or transit the frame is accomplished as follows:

- If the frame DA matches the set of MAC addresses assigned to given instance of the MAC, the frame is stripped from the ring.
- If the frame DA is a broadcast or multicast address:
  - If TTL > 1, the frame is both received and copied for transit.
  - If TTL = 1, the frame is received and stripped.
- If the frame SA matches this RPR MAC address, then the frame is discarded.

- If the frame has a bad CRC in the RPR MAC header, the frame is stripped and discarded. A bad CRC counter is incremented. These errors are accounted for as signal degradation on the upstream link.
- If the DA MAC address of the incoming frame does not match the set of MAC addresses assigned to this instance of the MAC and if the TTL field in the header of the frame is TTL <= 1 the frame is discarded.
- Finally, if none of the above conditions matches, the TTL field in the RPR MAC header is decremented by one, and the frame is transited.

The earliest time that the RPR MAC can decide whether to receive and remove or receive and copy the frame is upon reception of the entire RPR header (which is protected by the HEC). It does not wait for the entire frame to be received.

When the frame is received, a data.indicate signal is given to the MAC client to allow it to fetch the frame from the MAC.

6.3 Transmit Rules

Transmit frames are queued in the MAC client before they access the MAC entity. When a MAC client has a frame to send to the MAC entity, the MAC client indicates this by the DATA.request primitive. The MAC accepts the frame into the add path only if both of the following conditions are satisfied:

- There is no transit packet under transmission;
- The transit buffer is empty; and
- The media access rate control has not asserted a PAUSE.

More details about the PAUSE assertion are provided in the section on Media Access Control (Clause 11).

6.4 Scheduling

Transit frames are sent to the transit buffer. Frames are scheduled from the transit buffer by the arbiter that schedules between transit frames and transmit frames. The arbiter is a strict priority scheduler where the transit frames have highest priority.

The scheduling algorithm works as follows.

Step 1: Choose a frame to be transmitted using the following logic:

```
If (transit buffer has at least one byte)
   Choose a frame from the transit buffer
Else if (insert buffer has at least one frame)
   Choose a frame from the insert buffer
```

Step 2: Transmit the chosen frame with no pre-emption.

Step 3: Go back to Step 1.

In words, this works as follows:

- If there is a transit frame waiting, it is sent out right away to the outgoing link on a given ringlet.
- If there is no transit frame waiting, an add frame, if available, is sent. Thus the add frame waits until the transit buffer has no frame to send.
- Add frames cannot interrupt the transit frame under transmission (no preemption).
- Transit frames cannot interrupt the add frame under transmission (no preemption).
- In the store and forward mode of operation transit frames are received entirely before they are sent out.

6.5 Transit buffer considerations

Buffering of transit traffic may be done using a single transit buffer for all classes of traffic or multiple transit buffers. This architecture of the transit buffering is independent of the MAC fairness algorithm described in Clause 12.

6.6 Optional modes of operation

6.6.1 Cut-through mode

In the cut-through mode of operation, frame transmission can begin before it is entirely received. At a minimum, the RPR header should be received before beginning transmission to the outgoing ringlet, since the header has to be processed following the rules described in Section 6.2. If the RPR MAC is sending out an Add frame, while receiving a Transit frame, the latter will be stored until the transmit frame has been completely sent out. Only a single MTU of transit buffer is required in the transit path.

The advantage of this option is that it reduces the delay that frames experience in the transit path. In this mode of operation transit packets with an FCS error will be propagated. However, the errors can be detected and counted in the error statistics for the transit path as the frame is passed through.

6.6.2 Store and forward mode

In the store and forward mode of operation transit frames are received entirely before they are considered for transmission. This mode of operation allows a frame with an FCS error to be discarded and the transit error counter incremented. This mode of operation requires a minimum of two MTU transit buffer in the transit path.

The advantage of this option is that it eliminates damaged frames already in transit, so that they do not take up the bandwidth up to their destination in the rare instance of FCS error.

6.7 Transit path design in promiscuous mode of operation

See Section 6.4.1 of Darwin.

6.8 Circulation packet detection (stripping)

See Section 6.5 of Darwin.

6.9 Wrapping of data

See Section 6.6 of Darwin.

6.10 Pass-thru mode

See Section 6.7 of Darwin.

11. Media Access Control

11.1 Objectives

11.1.1 The need for access control

For a shared ring network in which the 802.17 MAC will be used, each ring segment carries traffic source by many stations. Each segment attached to a MAC (a station) carries both local client traffic and the traffic from the clients of other MACs upstream. Unless the upstream MACs control their access rates, their traffic may consume more than their fair share of the ring segment bandwidth. This could lead to congestion thereby blocking the local client from gaining access to the media.

Hence, the 802.17 MAC needs to employ some form of congestion avoidance mechanism to prevent congestion *before* it occurs. Congestion avoidance must be a proactive mechanism; it should not wait until congestion takes place before reacting. Such a mechanism must constantly monitor and control the access rate from each upstream MAC as well as from its local client. The congestion avoidance mechanism proposed for the 802.17 MAC is specified in Clause 12.

The congestion avoidance mechanism proposed has two phases: bandwidth allocation and access rate control. Bandwidth allocation is carried out to ensure fair access to each ring segment and more importantly to allow maximum utilization of all ring segments. The MAC directly attached to a ring segment is responsible for allocation of the segment's bandwidth between its local client and the clients of upstream MACs. The allocation should be fair (fairness criteria are defined in Clause 12) and efficient. In conjunction with the bandwidth allocation, each MAC must implement an access rate control to regulate the traffic from its attached client that the MAC allows on to the ring. The access rate control prevents upstream MACs from gaining access to the ring more than their allocated rates, creating congestion at a down stream ring segment. The access rate control proposed is specified in Clause 12.

11.1.2 Bandwidth allocation policy

Similar to any other resource allocation, the bandwidth allocation must be governed by a policy. In order to support service level agreement (SLA) services and to allow traffic engineering over 802.17 networks, the bandwidth allocation policy must contain essential capabilities so as not to impede its clients from implementing these services. Such capabilities include:

1. Supporting committed bandwidth provisioning for each station.
2. Supporting weighted fair allocation of available bandwidth for each station in addition to its committed bandwidth.
3. Supporting separation based on class of service, where the available bandwidth for a class is allocated to the members of that class first.

In a simple terms, the steps to determine bandwidth allocation for each ring segment are as follows:

1. Determine the amount of committed bandwidth currently used by each MAC on the ring.
2. Calculate the amount of unused committed and unused uncommitted bandwidth, namely available bandwidth.
3. Allocate available bandwidth fairly according to a pre-configured weight associated with each station on the ring.

With the ability to take committed (reserved) bandwidth into account, the total allocated bandwidth to each MAC is simply: *Total allocated bandwidth = committed bandwidth + weighted fair allocation of available bandwidth.*

Committed bandwidth is a static value; however a node needs to be aware of the committed bandwidths of all nodes that source traffic passing through it. Thus only the available bandwidth allocation needs to be

communicated to the sending MACs around the ring. The details of the allocation method are specified in Clause 12.

11.1.3 Independent access control for maximum spatial reuse

In order to achieve maximum bandwidth capacity of the ring, the access rate control must take into account the spatial reuse property of the ring that the 802.17 MAC seeks to exploit. The spatial reuse property is realized when a MAC receives and removes frames destined for it from the ring, freeing up the ring bandwidth so that the other MACs downstream can insert their own traffic (i.e., reuse the bandwidth).

The access rate control, therefore, must be capable of making an intelligent decision to not restrict spatial reuse when enforcing access to the ring. The client of an 802.17 MAC may send traffic to multiple destinations traversing many ring segments. If the MAC does not allow an independent access rate per destination, it is possible that the MAC sets the access rate low to satisfy the bandwidth allocated by one congested destination and severely limits the access rates to other uncongested destinations.

11.1.4 Virtual output queueing (or virtual destination queueing)

Another problem, somewhat related to the one discussed above, that the 802.17 MAC needs to prevent is head of line (HoL) blocking. This would normally occur in a MAC that can support only single FIFO access. If the MAC client implements a single FIFO to buffer frames awaiting access on to the ringlet, it is possible that a frame destined to an uncongested destination may be waiting behind an HoL frame that cannot gain access to the ring because its destinations is congested. Until the HoL frame is removed from the FIFO all frames behind are blocked, effectively preventing the local client from taking advantage of 802.17's spatial reuse property.

A well-known solution to this HoL blocking problem is a virtual output queue (VOQ) implementation. For VoQ, the 802.17 MAC must allow independent access per destination and inform its client of each access rate. The MAC client can then implement VOQ by maintaining a dedicated FIFO for each destination. With a per-destination buffer, one frame cannot be blocked by another frame for a different destination, hence eliminating HoL blocking completely and achieving 100% spatial reuse.

Note: Virtual output queueing is the same concept referred to as virtual destination queueing (VDQ) in the "Darwin" proposal.

11.2 The MAC Model

The MAC sublayer defines a media-independent facility, built on the media-dependent physical facility provided by the Physical Layer, and under the access-layer-independent LAN LLC sublayer (or other MAC client). It is applicable to a shared media resilient packet ring (RPR) network.

The LLC sublayer and the MAC sublayer together are intended to have the same function as that described in the OSI model for the Data Link Layer alone. In an RPR network, the notion of a data link between two network entities does not correspond directly to a distinct physical connection. Nevertheless, the partitioning of functions presented in this standard requires four main functions generally associated with a data link control procedure to be performed in the MAC sublayer. They are as follows:

1) Data encapsulation (transmit and receive)
    a) Framing (frame boundary delimitation, frame synchronization)
    b) Addressing (handling of source and destination addresses)
    c) Error detection (detection of physical medium transmission errors)

2) Frame handling
   a) Header processing
   b) Frame add-drop
   c) Transit path
3) Media access management
   a) Media access rate control (congestion avoidance)
   b) Link bandwidth allocation (congestion avoidance)
   c) Frame insertion (contention resolution)
4) Protection switching

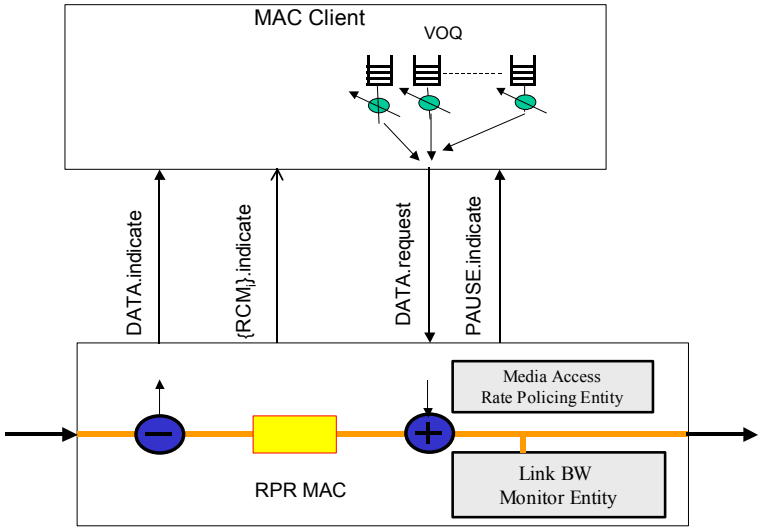

Figure 11-1 MAC model

Figure 11-1 is a simplified view of the 802.17 MAC. The figure shows how the MAC inserts frames onto the ring, transits frames through the ring, and receives frames from the ring. The set of service primitives shown are the ones required for the MAC and its client to communicate for the purpose of sending and receiving frames. The primitives are defined in Clause 5 (MAC Reference and Service Model). This set of primitives is provides the MAC client with sufficient information so that buffering of frames waiting for access to the ring can be maintained in the MAC client and not in the MAC. This is a highly desirable property considering that the 802.17 MAC is expected to scale in speed with advances in transmission technologies. More importantly, it allows the MAC to support a wide range of clients--from a dumb client having only a single FIFO to a client with advanced queueing features.

Once accepted by the MAC, the frame is said to be already on the ring for a media accessing purpose. The MAC, however, may implement one or more staging buffer/s to temporarily store accepted frames while it waits for an opportunity to place them onto the ring.

11.3 MAC Architecture

At a high level, the 802.17 MAC tasks are divided into two main functions. The first is the frame handling and transit path function that is specified in Clause 6 (MAC data path). The second is the bandwidth man-

agement function that covers bandwidth allocation and access rate control and is specified in Clause 12 (MAC Fairness).

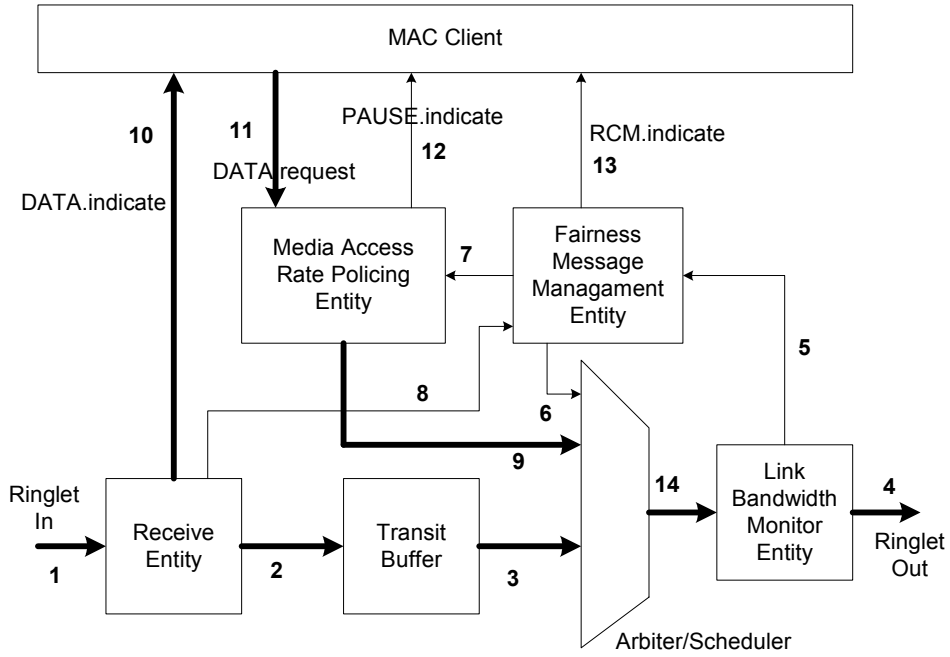

Figure 11-2: MAC architecture. The data path is shown in bold lines.

A detailed internal architecture of the MAC is shown in Figure 11-2. The blocks depict entities that perform specific functions within the MAC and are explained below. The numbers next to the arrows denote the flow of information between the various entities and will be explained later. The MAC is comprised of the following entities.

The *receive entity* examines the header of an 802.17 frame to determine whether the frame is to be received and removed, received and copied, or simply transited to the next MAC on the ring. Control frames such as those required by the MAC fairness algorithm are also identified and sent to the appropriate processing module.

The *transit buffer* is a buffer whose function is to hold transit frames as they wait for the insertion of a pending frame to complete. As mentioned earlier, the transit buffer may be implemented in a variety of ways ranging from a single transit buffer for all traffic to multiple buffers that are used for buffering transit traffic by class of service.

The *arbiter/scheduler entity* resolves contention among frames from the transit buffer, frames from the client inserted by the media access rate policing entity, and control frames from the fairness message management entity attempting to access the outgoing ring segment at the same time. The arbitration method shall be as specified in Clause 6 (MAC data path).

The *link bandwidth monitor entity* tracks the rate of traffic going through the outgoing ring segment for all stations including itself. Based on its measurement, the entity allocates available bandwidth in a weighted fair manner. The entity generates a rate control message periodically to update current bandwidth allocation. (See Clause 12 for more information about this.)

The *fairness message management entity* is responsible for sending and receiving fairness control messages to and from the other stations. This entity also responsible for passing the fairness information to the MAC client in cases where the client implements VOQ or some other form of queueing that may benefit from the knowledge of congestion on multiple links.

The *media access rate policing entity* uses the fairness messages to limit the access rate to the media so that traffic from the local client shall never exceed its allocated bandwidth on any ring segment. The PAUSE.indicate is used to signal the client that the MAC currently will not accept any frame from the client as it has exceeded its allocated rate.

The flow of information between the various entities is shown in the arrows. In the following text, refer to the figure for the numbers against the arrows for each of the information flows.

1. Frames are received and passed to the receive entity for processing.

2. After the receive entity has decided which frames are transit frames, the transit frames are passed to the transit buffer where they are queued and scheduled for transmission by the arbiter.

3. The arbiter schedules the add/transit frames and sends them out on the ringlet using the transmit rules specified in Clause 6.4.

4. Frames exit the link bandwidth monitor entity and are sent out on the ringlet.

5. Information about the fair rate for the local outgoing segment is sent to the fairness message management entity.

6. The fairness message management entity sends fairness messages to both ringlets. (The figure shows only one ringlet.)

7. The fairness message management entity supplies fairness information received from all of the nodes to the media access rate policing entity.

8. The fairness messages (for this ringlet) that are received are sent to the fairness message management entity for processing. Note that fairness messages will be received from both ringlets. The figure is simplified to show only a single ringlet.

9. The media access rate policing entity accepts the traffic and places it in a small insert buffer awaiting transmission by the arbiter.

10. For frames that are received and need to be passed on to the local client, a DATA.indicate primitive is used to inform the MAC client about the frame.

11. When the MAC client wants to transmit data, it indicates this using the DATA.request primitive.

12. The PAUSE.indicate primitive indicates to the MAC client that no more data will be accepted as the client has exceeded its allowable rate. Note that as a part of the PAUSE.indicate, the client must also be supplied with the segment number for which it is being paused.

13. The MAC supplies the MAC client with all information from the fairness messages in case they are able to make use of the individual fairness information received for policing/shaping.

14. The arbiter schedules frames for transmission. As the frames are transmitted, the link bandwidth monitor entity maintains per-source statistics by keeping a count of the number of bytes transmitted by each source on this segment.

12. MAC Fairness

MAC Fairness is achieved by the bandwidth management entity. This clause provides detailed requirements and specifications for the bandwidth management entity for fair media access.

12.1 Objectives of bandwidth management

Bandwidth management is a required MAC function because the media is a shared media. Due to station locality separation, an access method must be implemented to maintain and guarantee fair access to the media for all stations contending for the share resource--access bandwidth onto the medium.

The goals of bandwidth management are:

- Source-based weighted fair access to the ring.
- Maximize ring BW utilization.
- Maximize spatial reuse property of the ring.
- Support differentiated traffic types. Traffic of the same type from each station should experience identical performance while traffic from a different type may achieve better or suffer worse performance (performance metrics include throughput, delay, and jitter).
- Enable the virtual output queueing (VoQ) construct on the client that eliminates Head of Line (HoL) blocking due to station locality and traffic patterns (see Clause 11 for more detail).

12.2 Traffic types

The following describes traffic classes that must be supported by the MAC.

1) **Guaranteed class**. For this class of traffic, when a source (a MAC client) is not using all or some of its BW, the unused bandwidth must not be reallocated to any other source. No available bandwidth shall be allocated to this class. This class of service is typically used for circuit emulation.

2) **Committed class**. For this class of traffic, if a source is not using all or some of its committed (reserved) bandwidth, the unused bandwidth can be temporarily reallocated to other sources that need more bandwidth above their committed rates. This class can be sub-classified into in-profile or out-of-profile. In profile packets behavior like the guaranteed class and out of profile BW within the burst rate are shared with the opportunistic class.

3) **Opportunistic class**. This is analogous to the best effort class. The opportunistic class can also be considered as a special case of the committed class when a committed rate is set to zero (no rate is committed). Essentially, the opportunistic class relies solely on a (weighted or equal) fair share of available bandwidth.

Among all the stations on a ring, it is possible that stations may not fully utilize all the bandwidth allocated. They may not even have enough traffic to fully utilize their committed bandwidth. In order to maintain maximum utilization of each ring segment at all time, this unused bandwidth may be re-allocated to other stations that need more bandwidth. Hence, it is the task of the link bandwidth allocation entity to determine the demand of each upstream MAC and to provide instantaneous allocation of the bandwidth that becomes available.

12.3 Functional requirements for the bandwidth management entity

In order to achieve the goals specified in Clause 12.1, it is necessary that the 802.17 MAC have an effective means of reclaiming unused BW and redistributing the bandwidth fairly among sending stations. The following methods are required as a basic set of MAC functionalities in order to meet this requirement.

1. Calculation of the fair bandwidth allocation.
2. Signaling of the allocation to upstream MACs.
3. Policing the insert rate according to the allowed rate.
4. Detecting the onset of congestion.
5. Signaling to the MAC client for services that can be offered by a higher layer; e.g. virtual output queueing.

12.4 Components of the bandwidth management entity

The BW management entity consists of separate entities. These are as shown in Figure 11-2, and they are listed below along with the respective functions that they perform.

12.4.1 Link bandwidth allocation entity

The link bandwidth allocation entity monitors traffic use on the attached outgoing segment for both transit and ring ingress to determine the ring output link utilization. Traffic from each source MAC is tracked independently to determine the activity and rate at which each source MAC puts traffic on the segment. The bandwidth allocation method within this entity relies on this information to allocate bandwidth fairly based on a given weighted allocation method.

The bandwidth allocation procedure allows other MACs on the same ringlet to utilize the segment bandwidth fully under all traffic patterns. This must be achieved without violating guaranteed and committed bandwidth allocation and in accordance with the weight assigned to each station.

Upon receiving a new rate control message (RCM), each MAC simply stores the value contained within the message. This value is the RCF (rate control factor). The RCF when multiplied with the local node's weight provides the bandwidth allocated to this node by the segment that sourced the RCM.

Stations that do not wish to participate in available bandwidth allocation can set their weights to zero. In this case, the stations can always access the media up to, but not above, their committed rates.

The MAC algorithm to calculate the bandwidth allocation is a per-source leaky bucket approach that works in a similar fashion as the well-known weighted fair queueing (WFQ). Per-source byte counters track the amount the MAC receives from each remote source including its own client.

The bandwidth allocation algorithm is a simple two-step process that is executed once every calcInterval. It first determines the AvailableRingBW by subtracting out the Guaranteed Class BW from the total link BW. Next, the RCF is obtained by dividing (AvailableRingBW – sum of the committed rate of all active sources) by the total weight of all active stations.

It is worth noting that the calculated values are instantaneous ones that can fluctuate greatly from one calcInterval to the next depending on the traffic pattern. Thus the value of the RCF needs to be smoothed out before sending to the other MACs on the ring. The pseudo-code in Annex J uses a standard low-pass filter to smooth the value of the RCF.

The pseudo-code for a sample implementation that computes the fair rate in accordance with these rules is provided in Annex J.

12.4.2 Fairness message management entity

The fairness message management entity is responsible for sending and receiving rate control message (RCM) frames to and from the other stations. This entity is also responsible for passing the RCMs to the MAC client in cases where the client implements VOQ or some other form of queueing that may benefit from the knowledge of congestion on multiple links.

12.4.3 Media access rate policing entity

The media access rate policing entity is responsible for policing the ring access for packets for different classes. It also performs the execution of fair access on to the ring as dictated by the fairness algorithm. Finally, it signals the MAC client sub-layer with RCF information for VoQ support

12.4.3.1 Media access rate control

IEEE 802.17 MAC controls access to the media on a per-segment basis. Earlier, this clause has described how the bandwidth management entity determines the maximum allowed rate on a per-segment basis around the ring. This section describes how Media Access Control limits the access into the media.

There are two type of MA_CONTROL.indication that the MAC conveys to the MAC clients as follows:

- **Pause signal:** The pause signal in the MA_CONTROL.indication frame indicates which link segment's bandwidth is violated. Thus the pause signal also indicates the congested segment number.

- **RCM signal:** The RCM messages are constantly indicating to the MAC the rate available on a per-segment basis. The MAC client can also use the RCM information to shape the traffic locally sourced onto the media so that the MAC rate control doesn't ever need to send pause signals.

The pause signal and RCM signals overlap in terms of functionality, but they are both necessary to be able to serve traditional clients that cannot interpret the RCMs as well as those clients that can take advantage of the information provided in the RCMs.

12.4.3.2 Policing method

The policing method used is credit based. The policer's role is to monitor the insert rate from the MAC client to all ring segments and enforce compliance, prohibiting the client from exceeding its allocated rate of access. For each downstream segment that the client's inserted traffic has to traverse, the media access rate control entity maintains an associated credit counter.

At every T_{update} (usec) interval, credits are added to each counter based on the bandwidth allocation received for the segment derived by multiplying the current RCF with the node's weight. The number of credits loaded into the counter divided by T_{update} is equal to the allowed rate that the client can insert traffic through that segment.

To support VOQ, for each frame received from the client, a number of credits equal to the size of the packet are deducted from each segment (including the local one) that the packet will traverse.

The peudo-code for a sample implementation is provided in J.1.

12.4.3.3 Access control with a single rate policer

The single rate policer is supported by the 802.17 MAC as an optional capability. This capability enables the 802.17 MAC to inter-work with the simple local fairness control methods of pre-standard technologies. It supports pre-standard MACs that have only a single rate policer but wish to exploit local BW reuse of the ring to some degree.

The concept here is to find the most congested link in the topology *that appears before a node that advertises the MAX_RCF value in its RCM*. Then, only that rate is used for policing all of the traffic from this client.

The pseudo-code for a sample implementation of a MAC with a single rate policer is provided in J.2.

12.5 Fairness control message

The rate control message (RCM) is a means for a downstream MAC to signal to all of the upstream MACs the current allocation of available bandwidth of the attached outgoing segment that it gives to each of them. The message can be sent periodically to refresh the values or immediately when the onset of congestion is detected.

12.5.1 RCM format

The Bandwidth Management control message is called an RCM. This frame is broadcast to all stations on the ring. The RCF update control message contains the ringlet ID value octet and a 4-octet value for the rate control factor (RCF). The payload type indicator (PTI) value is set to indicate an RPR control frame (TBD). The TTL value is set to maxNodeCount. The COS value is set to 111 and the OC is set to 01-00. The RCF within the message applies to the downstream segment from the source of the message on the ringlet specified.

6 OCTETS	DA	
6 OCTETS	SA	
2 OCTETS	PTI= RPR CONTROL FRAME	
1 OCTET	TTL=maxNodeCount	
4 BITS	COS =111	RESERVED = 00000
2 OCTETS	OC = 01-00	
1 OCTET	RINGLET ID	
4 OCTETS	RCF VALUE	
4 OCTETS	FCS	

Figure 12-1: RCM frame format

12.5.2 Generation of fairness messages

RCM messages are periodically transmitted as a Transmit Timer Expires. This is a soft state operation that is highly robust. The recommended Transmit Timer value is 10 msec.

At each transmission time interval, a message is sent to the station upstream. For dual counter rotation ring the RCM is broadcasted on both rings.

12.6 Multicast support

Multicast traffic is expected to be source stripped. Hence, it is a special case of unicast flow control where the destination is the source node itself.

12.7 Bandwidth management in promiscuous mode

MACs operating in promiscuous mode will generate RCM messages that contain an RCF value equal to the outgoing segment link rate of the upstream node on the ring. RCM messages will advertise the entire outgoing link segment rate to all other nodes.

Media Access Rate control polices only traffic that originates from its local client on a per-destination segment basis using the RCFs received from each segment.

A ring containing some nodes operating in promiscuous mode while others are operating in a normal mode would operate as follows. A MAC operating in normal mode will advertise the RCM message for the fair share of attached outgoing link segment. A receiving MAC in promiscuous mode will multiply the RCF with the local weight to determine the fair add rate for that link segment for the purpose of policing.

Annex J Code Examples

This annex contains the pseudo-code for a regular 802.17 MAC and a MAC with only a single rate policer.

J.1 Bandwidth management for the MAC

This algorithm is executed on a per-ringlet basis.

J.1.1 Variables and constants used

Constants/configured values

- curr_node: Identifies the node executing the algorithm.
- calcInterval: The interval at which the fair share is calculated periodically. A smaller value of the interval makes the algorithm more responsive, but less stable. For 10G rings, the recommended value is 100 usec. A larger value may be used for slower link speeds.
- Tupdate: Time interval at which credits are recomputed for each segment. Again this is tradeoff between stability and responsiveness. The recommended value for a 10G ring is 100 usec. A larger value may be used for slower link speeds.
- RCM_send_interval: The interval between sending of RCM messages from a node. A new RCM message is sourced every RCM_send_interval. The recommended value for a 10G ring is 10 msec.
- W[i]: The weight for node i used for proportional sharing of any excess bandwidth on the ring. Source i will receive W[i]/(sum_i W[i]) proportion of the available bandwidth.
- AvailableRingBW: The amount of bandwidth available on the outgoing link segment at the curr_node after subtracting the committed bandwidth of the guaranteed flows that pass through it or originate at it.
- MAX_CREDIT: A maximum threshold to prevent segment credits from accumulating without bound when this node is not using all of its allocated share on that segment. It should be set to 1 or 2 MTUs.

Variables

- source_node: Identifies the node that sourced the packet on the ring.
- packet_length: the length of the packet in bytes.
- Bucket[i]: The number of bytes received from node i.
- RCF[i]: The rate control factor for node i (or segment i). This information is available for all segments through the RCMs received.
- R[i]: The reserved rate for traffic sourced by node i.
- RCF_sampled: Rate Control Factor - this measures the normalized rate at which each source is allowed to send traffic through the node executing this algorithm. For any other node to determine its share on the current node's segment, it must multiply the RCF by its weight.
- F: The allowable rate for the curr_node on node (or segment) on the ring. This is a temporary variable.
- segment_credit[i]: The number of bytes that this node is allowed to transmit on segment i. This value accumulates based on the allowable rate, but and is reduced whenever a packet is transmitted that would use that segment. If negative, it indicates that the node has over-subscribed its budget for that segment.
- segment_paused[i]: When curr_node exceeds its allocated capacity on segment i, this variable is set to TRUE.
- NumNodes: The number of nodes on the ring.
- RCF_new: A global variable that stores the RCF used and advertised by curr_node.
- RCF_old: A temporary variable.

J.1.2 Traffic monitoring

```
@packet arrival (from the ring or from the local client)
    bucket[source_node] += packet_length;
```

J.1.3 Action on receipt of a fairness message

```
@receipt of fairness message
    /* update the RCF for the node that sent the fairness message */
    RCF[source_node] = Packet.RCF;
```

J.1.4 Computation of the fair share at the local node

```
@calcInterval
    /* initialization */
    sum_R = 0;
    sum_W = 0;

    /* do for each source node */
    for (i = 0; i < NumNodes; i++) {

        if (bucket[i] > 0) { /* do only for active flows */
            sum_R += R[i];
            sum_W += W[i];
        }

        /* drain the bucket for node i */
        bucket[i] -= calcInterval * (R[i] + W[i] * RCF_new);

        if (bucket[i] <= 0) {
            bucket[i] = 0;
    }
    RCF_sampled = min(((AvailableRingBW - sum_R)/ sum_W), AvailableRingBW);

    /* filter the newly sampled value */
    RCF_old = RCF_new;
    RCF_new = a * RCF_old + (1 - a) * RCF_sampled;
```

Because the number of flows active or inactive is a discrete quantity, the sampled RCF will likely end up varying significantly across calcIntervals. To smooth the value of the RCF advertised to all of the nodes, we use the low pass filter shown above. The value of `a' used is 0.95. In this way, the value of RCF_new is influenced by both RCF_old and RCF_sampled providing a smoothed value. Note that RCF_new is the value that is advertised to all nodes on the ring and is the value that should be used for draining the bucket during the next calcInterval.

J.1.5 Sending of RCM messages

```
@RCM_send_interval
    Packet.RCF = RCF_new;
    Send RCM Packet;
```

J.1.6 Access control

```
@Tupdate
    for each (link segment i on the ring) {

        /* calculate the allowable bandwidth for this node */
        F = R[curr_node] + W[curr_node] * RCF[i];

        /* accumulate credits for each segment */
        segment_credit[i] += F * Tupdate;
        segment_credit[i] = MIN(segment_credit[i], MAX_CREDIT);
```

```
        /* has client has exceeded credit for segment? */
        if (segment_credit[i] < 0) {
            segment_paused[i] = TRUE;
            assert PAUSE.indicate for segment i;
        } else {
            segment_paused[i] = FALSE;
            clear PAUSE.indicate for segment i;
        }
    }

@DATA.request from client
    for each (link segment i between source and destination) {
        if (segment_paused[i] == TRUE) {
            reject DATA.request;
            return;
        }
    }
    accept DATA.request;
    for each (link segment i between source and destination) {
        /* deduct segment credit */
        segment_credit[i] -= packet_length;
    }
```

J.1.6 Implementation complexity

The following is the memory requirement:

- R[]: 256 nodes x 32 bits = 8 kbits (provisioned)
- W[]: 256 nodes x 8 bits = 1 kbits (provisioned)
- Buckets: 256 nodes x 32 bits = 8 kbits
- RCF[]: 256 nodes x 32 bits = 8 kbits
- segment_credit[]: 256 nodes x 16 bits = 4 kbits
- ~40K gates for the link bandwidth monitor entity (monitoring and fair rate computation)
- ~30K gates for the media access rate policing entity (access control policing)

J.2 Bandwidth management for a MAC with a single rate policer

This algorithm is executed on a per-ringlet basis. The algorithm is very similar to the one in the previous section, but is developed for clients that are assumed to have only a single queue and that are not capable of maintaining per-destination queues.

The primary difference with the previous algorithm is that at the time of receiving an RCM, or at the time of computing the RCF of the local node, the minimum of all the RCFs until an uncongested link is encountered is stored. That value, RCF_min, is the only one used by the policer for access control.

J.2.1 Variables and constants used

Constants/configured values

- curr_node: Identifies the node executing the algorithm.
- calcInterval: The interval at which the fair share is calculated periodically. A smaller value of the interval makes the algorithm more responsive, but less stable. For 10G rings, the recommended value is 100 usec. A larger value may be used for slower link speeds.

- Tupdate: Time interval at which credits are recomputed for each segment. Again this is tradeoff between stability and responsiveness. The recommended value for a 10G ring is 100 usec. A larger value may be used for slower link speeds.
- RCM_send_interval: The interval between sending of RCM messages from a node. A new RCM message is sourced every RCM_send_interval. The recommended value for a 10G ring is 10 msec.
- W[i]: The weight for node i used for proportional sharing of any excess bandwidth on the ring. Source i will receive W[i]/(sum_i W[i]) proportion of the available bandwidth.
- AvailableRingBW: The amount of bandwidth available on the outgoing link segment at the curr_node after subtracting the committed bandwidth of the guaranteed flows that pass through it or originate at it.
- MAX_CREDIT: A maximum threshold to prevent segment credits from accumulating without bound when this node is not using all of its allocated share on that segment. It should be set to 1 or 2 MTUs.

Variables

- source_node: Identifies the node that sourced the packet on the ring.
- packet_length: Length of packet that is to be transmitted.
- cong_node: The most congested node (or segment) along the ring until we either get back to the source or we hit an uncongested node (one whose RCF is MAX_RCF).
- RCF[i]: The rate control factor for node i.
- R[i]: The reserved rate for traffic sourced by node i.
- NumNodes: The number of nodes in the topology.
- F_cong: The allowable rate for this node on the congested link.
- RCF_min: Rate control factor for the congested segment. This value is derived when an RCM is received.
- segment_credit_cong: The number of bytes that this node is allowed to transmit on the congested segment. This value accumulates based on the allowable rate, but and is reduced whenever a packet is tranmitted. If negative, it indicates that the node has over-subscribed its budget.
- segment_paused_cong: When curr_node exceeds its allocated capacity on the congested segment this variable is set to TRUE.

J.2.2 Traffic monitoring

```
@packet arrival (from the ring or from the local client)
    bucket[source_node] += packet_length;
```

J.2.3 Action on receipt of a fairness message

```
@receipt of fairness message
    /* update the RCF for the node that sent the fairness message */
    RCF[source_node] = Packet.RCF;

    /* assume nodes are sorted by distance from this node on the ringlet */

    RCF_min = RCF[0];
    R = R[0];

    /* is the sending station within the span? */
    for (i = 0; i < NumNodes; i++) {

        /* if I hit a node that sends a MAX_RCF, then I don't
           contribute to congestion downstream of that */

        if (RCF[i] == MAX_RCF) {
            break;  /* get out of the for loop because we are done */
        }

        /* is this link more congested than any other so far? */
        if (RCF[i] < RCF_min) {
```

```
            RCF_min = RCF[i];
            R_cong = R[i];
        }
    }
```

J.2.4 Computation of the fair share at the local node

```
@calcInterval
    /* initialization */
    sum_R = 0;
    sum_W = 0;

    /* do for each source node */
    for (i = 0; i < NumNodes; i++) {

        if (bucket[i] > 0) {  /* do only for active flows */
            sum_R += R[i];
            sum_W += W[i];
        }

        /* drain the bucket for node i */
        bucket[i] -= calcInterval * (R[i] + W[i] * RCF_new);
        if (bucket[i] <= 0)
            bucket[i] = 0;

    }
    RCF_sampled = min(((AvailableRingBW - sum_R)/ sum_W), AvailableRingBW);

    /* filter the newly sampled value */
    RCF_old = RCF_new;
    RCF_new = a * RCF_old + (1 - a) * RCF_sampled;

    /* check if we are lower than RCF_min that was computed when a fairness
       message was received */
    if (RCF_new < RCF_min) {
        RCF_min = RCF_new;
    }
```

J.2.5 Sending of RCM messages

```
@RCM_send_interval
    Packet.RCF = RCF_new;
    Send RCM Packet;
```

J.2.6 Access control

```
@Tupdate
    /* calculate the allowable bandwidth for this node */
    F_cong = R[curr_node] + W[curr_node] * RCF_min;

    /* accumulate credits for each segment */
    segment_credit_cong += F_cong * Tupdate;
    segment_credit_cong = MIN(segment_credit_cong, MAX_CREDIT);

    /* has client has exceeded credit for segment? */
    if (segment_credit_cong < 0) {
        segment_paused_cong = TRUE;
        assert PAUSE.indicate;
    } else {
        segment_paused_cong = FALSE;
        clear PAUSE.indicate;
    }
```

```
@DATA.request from client
    if (segment_paused_cong == TRUE) {
        reject DATA.request;
        return;
    }
    accept DATA.request;
    segment_credit_cong -= packet_length;
```

Annex K Implementation Guidelines

K.1 MAC client options

This annex shows the various types of MAC clients that can be supported if the MAC conforms to the clauses specified in this document.

The MAC can serve multiple types of clients with no additional support by the clients:

- Traditional Clients
  1. LLC client
  2. Bridging client
  3. Routing client
- Enhanced client with VoQ functionality

Traditional LLC clients, bridging and routing clients do not implement virtual output queueing to avoid head of the Line blocking problem. The IEEE 802.17 MAC should be backward compatible to serve the traditional clients as well as new clients that implement VoQ functionality.

K.1.1 Single output queueing: Traditional MAC client

Traditional clients implement a single queue structure for all destinations at shown in Figure K-1. Since the MAC client doesn't implement per destination virtual output queueing to avoid head of line blocking, it can completely ignore the RCM messages. Traditional MAC clients only need to be aware of PAUSE assertion to stop frames from accessing the MAC. Traditional MAC clients can also ignore the information about the *most congested link.*

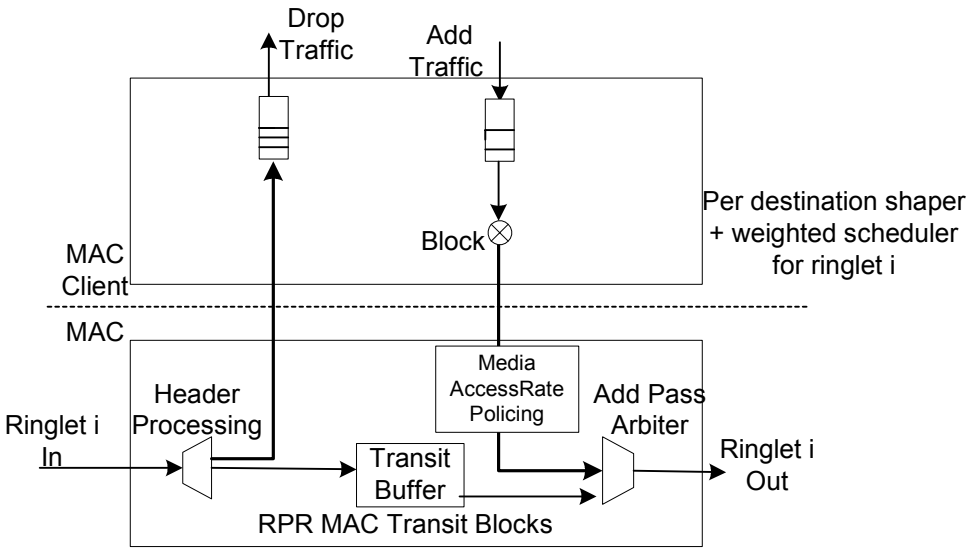

Figure K-1: A single queue with a traffic shaper in the add path

K.1.2 MAC client with virtual output queueing

One of the objectives of RPR is to maximize the spatial reuse in the RPR ring; i.e. to maximize the link utilization for packet flows with arbitrary (source, destination) pairs. When a single node is sending traffic

to two or more nodes, head of line blocking occurs if packets bound for multiple destinations are queued in a single add queue that has been blocked for at least one of the destinations due to ring fairness messages.

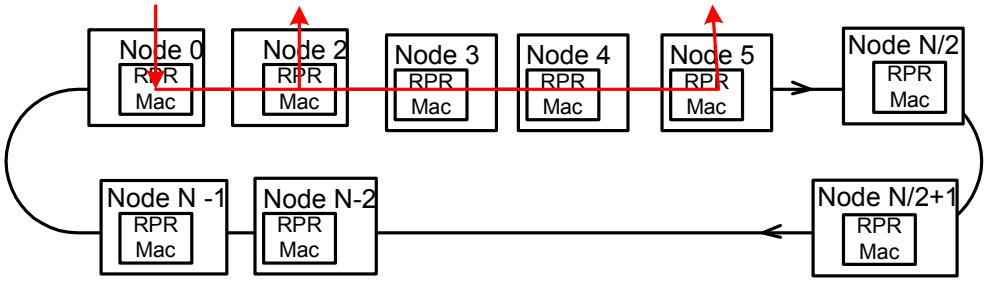

Figure K-2: Ring illustrating the head of line blocking problem

Head of line blocking can be explained very simply. In the ring shown above, Node 0 has two aggregate flows to send to Node 2 and Node 5. This is a common scenario for many applications such as transparent LAN services and peer-to-peer routing.

If the link between the Node 4 and Node 5 is congested, Node 4 will send the appropriate congestion avoidance/rate shaper parameter information to Node 0. This will slow or block the packets destined for Node 5. When packets destined for Node 5 reach the head of the queue, the packets destined for Node 2 will be slowed or blocked by the packet(s) destined for Node 5. This is head of line blocking. This problem has been long addressed in high-speed crossbar switches by the use of virtual output queueing.

K.1.2.1 MAC client with VoQ and MAC-based policing

The proposed 802.17 MAC clients solve the above problem through the use of virtual output queueing (VOQ), where there is an output queue for each destination node as shown in Figure K-3. The pause signal includes the most congested destination link. Thus MAC client needs stop all the destination flows that are traversing through that congested link. Mac client can get this topology related information from the LME where the current topology database information is stored.

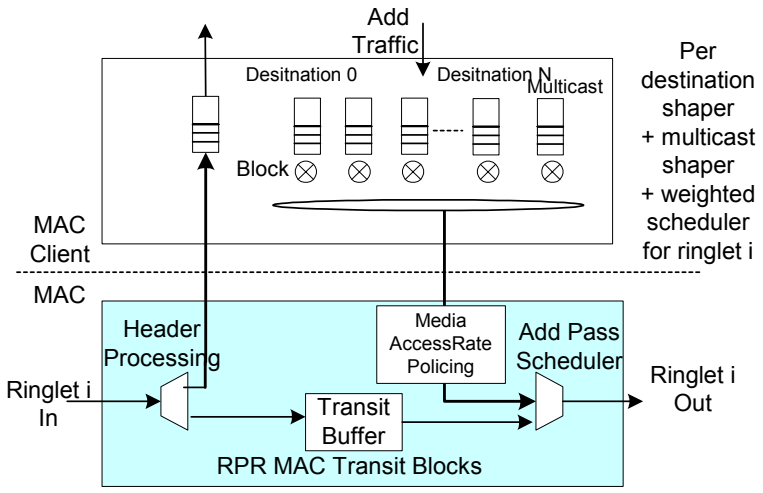

Figure K-3: Virtual output queueing in the MAC client and media access control using pause signaling

K.1.2.2 MAC client with VoQ and client-based shaping

In addition to the Pause messages MAC client relays RCM messages to the MAC client. RCM messages propagate accurate congestion information for each link. RCM messages provide the amount of bandwidth available in each destination link segment. Given the bandwidth information for each link segment, and using the information from LME (Layer Management Entity), the MAC client can assign bandwidth to each flow in a fair manner for each link according client system policy.

Now having determined allowed bandwidth on per destination basis, MAC client can shape the traffic on per destination queue basis as shown in Figure K-4. This avoids Pause signaling from the MAC. This enables better shaping of streams into the MAC increasing the link utilization.

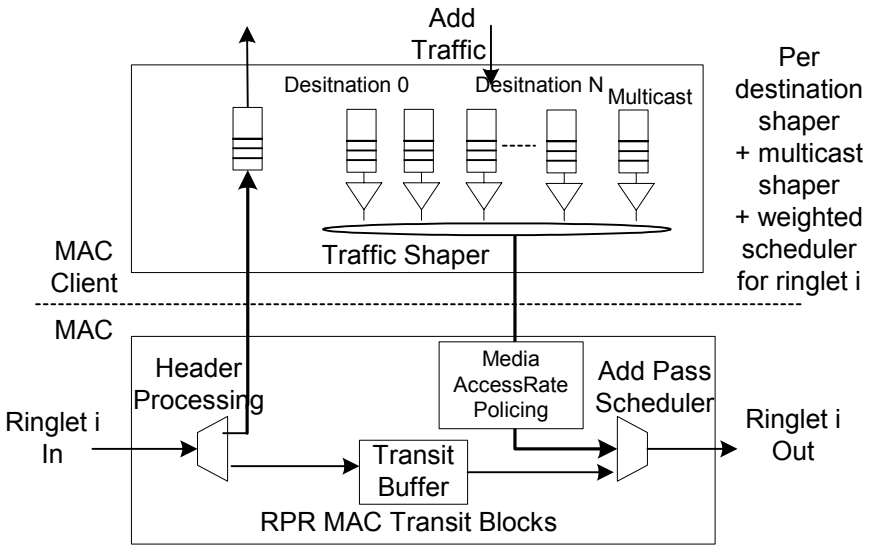

Figure K-4: Virtual output queueing with shaping in the MAC client

K.1.3 MAC client with class of service (CoS) support

The 802.17 MAC client can support a differentiated services architecture that specifies up to eight classes. Once an incoming packet is classified, marked, and policed according to the relevant SLA, these packets only need to be sorted into 8 class-based queues. In the MAC Client shown in Figure K-5, packets are sent to 8 (or fewer than 8) class queues on a per destination basis (VoQ). The 8 class queues corresponding to a given destination are scheduled on a strict priority basis.

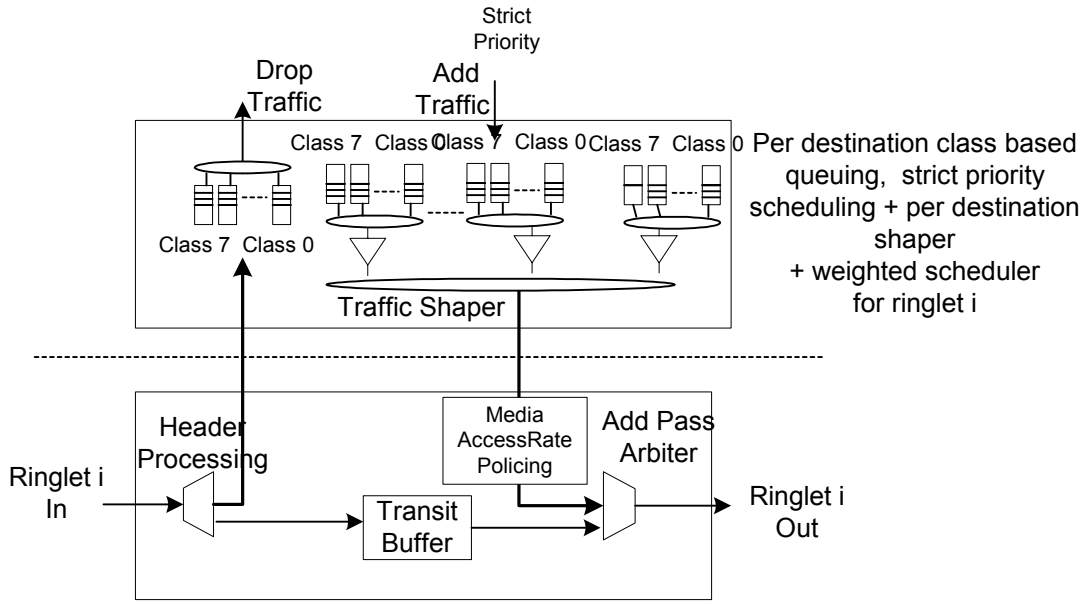

Figure K-5: Class of service based per-destination queueing and scheduling in the add path of the MAC client

As discussed earlier, the MAC bandwidth management supports three classes of services on the transmission medium:

- Guaranteed class
- Committed class
- Opportunistic class

Even though differentiated services can specify up to eight classes most of the packet networks services can be categorized into 4 classes as follows:

- Committed delay-sensitive service
- Committed delay-insensitive service
- Overcommitted provisioned delay-insensitive service
- Best-effort service

Committed delay-sensitive class can be mapped to the guaranteed class. Committed delay-insensitive service class can be mapped to committed class. Overcommitted and best-effort classes can be combined into opportunistic class.