9. MAC ~~fairness~~ fairness

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

References:
None.

Definitions:

Abbreviations:

Revision History:

Draft 0.1, February 2002	*Initial draft document for RPR WG review.*
Draft 0.2, April 2002	*Revised according to WG comments for TF review.*
Draft 0.3, June 2002	*Revised according to WG comments for TF review.*

9.1 Overview

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

Missing things in the overview section include:

- Functional block diagram and contents of clause
- Notational conventions

Detailed state diagrams are needed that describe:
the events that cause fairness messages to be sent;
the actions when a fairness message is received;
congestion and congestion subsided, etc.

The PICS needs to be added.

Need to make sure that reference model clause is in sync with this one.

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

Missing things in the overview section include:

- Functional block diagram and contents of clause
- Notational conventions

Detailed state diagrams are needed that describe:
the events that cause fairness control messages to be sent;
the actions when a fairness control message is received;
congestion and congestion subsided, etc.

The PICS needs to be added.

Need to make sure that reference model clause is in sync with this one.

9.1.1 Scope

~~This clause defines the fairness protocol for RPR MACs. The fairness protocol ensures a fair distribution of available bandwidth across all stations on a ring even during times when the ring is heavily utilized or congested. This clause specifies the MAC fairness protocol and how the MAC uses the protocol to enforce fairness among stations on the ring.~~

This clause defines the fairness algorithm for RPR MACs how the MAC uses the algorithm to enforce fairness among stations on the ring. The RPR fairness algorithm (RPR-FA) is a local fairness mechanism that enforces fairness among all the stations on a ring, even during times when the ring is heavily utilized or congested. Only fairness eligible traffic (the class-B traffic in excess of the provisioned amount, and all class-C traffic) is subject to the fairness algorithm.

9.1.2 Goals and objectives

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

We could not agree on a definition for stability, and hence the corresponding objective has been removed. If we find an acceptable way to describe stability, then we can add it later. For now, note that contributions are invited on a suitable definition (and corresponding objective) for stability. One possible definition is: ***Stability****—The protocol should not oscillate regardless of the type of traffic presented to the network.*

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

We could not agree on a definition for stability, and hence the corresponding objective has been removed. If we find an acceptable way to describe stability, then we can add it later. For now, note that contributions are invited on a suitable definition (and corresponding objective) for stability. One possible definition is:
Stability*—The protocol should not oscillate regardless of the type of traffic presented to the network.*

The fairness protocol has the following objectives:

a) **Source-based weighted fairness—**On any given ~~segment~~ link on the ringlet, the available bandwidth is allocated to each ~~node~~ station in proportion to its relative weight. For example, if every ~~node~~ station has an equal weight, then the available bandwidth on the ~~segment~~ link should be shared equally by all ~~nodes~~stations. On the other hand, if one ~~node~~ station has a higher weight, the bandwidth allocated to that ~~node~~ station should be in proportion to the ~~node~~station's weight divided by the sum of the weights of all the ~~other~~ stations.

b) **Reclamation of unused committed bandwidth**—The fairness protocol should be able to reclaim unused ~~bandwidth~~ bandwidth, including that which is ~~provisioned but not reserved~~provisioned.

c) **Support for ~~non-VDQ~~ single choke and ~~VDQ~~ multi choke capable clients**—The fairness protocol should be able to support clients whether they utilize a single choke point or ~~not they perform~~ multiple choke points (such as those clients performing virtual destination queueing (VDQ))~~. VDQ is a solution to the head-of-line (HOL) blocking problem~~.

d) **Fast response time**—Because data traffic tends to be bursty, in order to ensure maximum ring bandwidth utilization and to ensure that the protocol is responsive to instantaneous changes in traffic load, it must have a fast response time.

e) **High bandwidth utilization on the ring—**The protocol should be able to achieve very high levels of bandwidth utilization even under heavy loads approaching 100% of the ring capacity.

f) **Scalability**—The protocol should be scalable and should be able to function predictably for all ringlet speeds and ring diameters allowed by this standard.

9.1.3 Relationship to other clauses

~~TBD. Will be done by referring to the MAC reference model. Also need to refer to the appropriate primitives in the MAC service interface definitions.~~

The RPR-FA is implemented within a control entity called the Fairness Control Unit (FCU) located in the MAC Control Sublayer, as described in Clause 5.

The FCU uses the following variables defined in Clause 6: add_rate, add_rate_congested, fw_rate, fw_rate_congested.

The FCU provides the following variables for use in Clause 6: allowed_rate, allowed_rate_congested, TTL_to_congestion. The usage of these variables by Clause 6 is described in 9.10 .

9.2 Acronyms

This clause contains the following acronyms:

FA	fairness algorithm
FCM	fairness control message
FCU	fairness control unit
FE	fairness eligible
LR	line rate
MC	multi choke
RTT	round trip time
SC	single choke

9.3 Variables and terminology used

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

This section has been added by the editor. We need to have a list of definitions of all the variables used by the fairness algorithm which is completely missing from the present text.

This clause contains the following definitions and formula variables:

9.3.1 active_stations: Calculated. The number of stations that were active (as measured by having at least 1 transited fairness eligible frame) on the outgoing link of a ringlet during the last aging_interval.

9.3.2 add_rate: Calculated. A running byte count of all FE marked packets transmitted to the ringlet from the add queues of the local station during the previous AGECOEF aging_intervals (including the current interval), and smoothed over the previous AGECOEF aging_intervals. This rate is enforced by the Sd shaper to be less than or equal to the allowed_rate (+ 1 MTU - 1 byte).

9.3.3 add_rate_congested: Calculated. A running byte count of all FE marked packets transmitted to the ringlet from the add queues of local station during the previous AGECOEF aging_intervals (including the current interval), and smoothed over the previous AGECOEF aging_intervals, that have a destination past the congestion point. This rate is enforced by the Sc shaper to be less than or equal to the allowed_rate_congested (+ 1 MTU - 1 byte).

9.3.4 add_rate_congested_ok: Calculated. A boolean value indicating if the current add_rate_congested is within acceptable limits. This value is an input to the sendC signal defined in Clause 6.

9.3.5 add_rate_ok: Calculated. A boolean value indicating if the current add_rate is within acceptable limits. This value is an input to the sendC signal defined in Clause 6.

9.3.6 advertised_fair_rate: Calculated. The normalized rate (in bytes) advertised in an FCM giving the fair number of bytes that can be sent during the next aging_interval.

9.3.7 advertisement_interval: Configured. The interval at which FCMs are sent. This time is less than the aging_interval. The default value for this constant is a rate that uses .125% of the link bandwidth (e.g. 40 usec for a LINK_RATE of 2.5 Gbps).

9.3.8 AGECOEF: Configured. The coefficient used for aging the running add_rate, add_rate_congested, fw_rate, and fw_rate_congested. The default value for this constant is 4. Values other than factors of 2 are not required to be supported.

9.3.9 aging_interval: Configured. The interval at which aging and low pass filtering functions are performed. The value to use for each supported link speed is shown in Table 9.1.

Table 9.1—Rate Coefficient

Link Speed	aging_interval
OC-12 and higher rates	100 usec
OC-3	400 usec

9.3.10 aggressive: Configured. Aggressive at maximizing link utilization. Local station fairness (during transitions). Trades off stability for utilization (during transitions).

9.3.11 allowed_rate: Calculated. The rate at which the local station is allowed to transmit FE marked packets to the ringlet, specified as the number of bytes per AGECOEF aging intervals. During any aging_interval, the MAC may send up to allowed_rate minus add_rate FE bytes. This value is an input to the Sd rate shaper/limiter defined in Clause 6.

9.3.12 allowed_rate_congested: Calculated. The rate at which the local station is allowed to transmit FE marked packets to the ringlet that have a destination past the congestion_point, specified as the number of bytes per AGECOEF aging intervals. During any aging interval, the MAC may send up to allowed_rate_congested minus add_rate_congested FE bytes beyond the congestion_point. This value is the input to the Sc rate shaper/limiter defined in Clause 6.

9.3.13 congested: Calculated: The indication of whether the local station considers itself to be congested in trying to add traffic.

9.3.14 congestion_point: Calculated. The most congested station on the ringlet reporting a congestion condition (as indicated by the SA and TTL in the most recent SC-FCM).

9.3.15 conservative: Configured. Conservative at maximizing link utilization. Congested span fairness (during transitions). Trades off utilization for stability (during transitions).

9.3.16 fairness eligible: Definition. A quality of a packet that indicates (as specified in 8) if it is subject to the fairness algorithm. Packets that are subject to fairness are the class-B packets added in excess of the station's provisioned amount, and all class-C traffic. See also 6.

9.3.17 FULL_RATE: Constant. A special value for control_rate, indicating no congestion. The value for this constant is an all "1"s value.

9.3.18 full_threshold: Configured. The STQ threshold that indicates that the STQ of a dual-queue MAC is almost full. Same as NEEDY in Clause 6. The default value for full_threshold is STQ - 1 MTU.

9.3.19 fw_rate: Calculated. A running byte count of all FE marked packets transmitted to the ringlet from the transit queues of the local station during the previous AGECOEF aging_intervals (including the current interval), and smoothed over the previous AGECOEF aging_intervals.

9.3.20 fw_rate_congested: Calculated. A running byte count of all FE marked packets transmitted to the ringlet from the transit queues of the local station during the previous AGECOEF aging_intervals (including the current interval), and smoothed over the previous AGECOEF aging_intervals, that have a destination past the congestion_point.

9.3.21 high_threshold: Constant. The threshold that indicates that a MAC is experiencing some congestion. When the STQ depth or link utilization exceed this value, the outgoing link is considered congested.

9.3.22 LINK_RATE: Configured. Conceptually, the actual maximum transmission rate for the outbound link. For fairness formulas, it is interpreted as the maximum number of bytes that can be transmitted in AGECOEF aging_intervals. This rate is less than AGECOEF * aging_interval worth of bytes.

9.3.23 local_fair_rate: Calculated. The rate at which the local station may transmit FE marked packets, regardless of downstream congestion, specified as the number of bytes per AGECOEF aging_intervals.

9.3.24 local_station: Definition. The station in which the local MAC resides.

9.3.25 low_threshold: Constant. The threshold that indicates that a MAC may soon start experiencing congestion. When the STQ depth or link utilization exceed this value, the outgoing link is considered imminent to being congested.

9.3.26 lp_add_rate: Calculated. A low pass filtered value of add_rate.

9.3.27 lp_add_rate_congested: Calculated. A low pass filtered value of add_rate_congested.

9.3.28 lp_fw_rate: Calculated. A low pass filtered value of fw_rate.

9.3.29 lp_fw_rate_congested: Calculated. A low pass filtered value of fw_rate_congested.

9.3.30 lp_nr_xmit_rate: Calculated. A low pass filtered value of nr_xmit_rate.

9.3.31 LPCOEF: Configured. The coefficient used for low pass filtering the add_rate and fw_rate to result in lp_add_rate and lp_fw_rate. The default value for this constant is 64. Values other than factors of 2 are not required to be supported.

9.3.32 MAX_ALLOWED_RATE: Configured. The maximum value for allowed_rate. The default value for this constant is LINK_RATE.

9.3.33 multi choke: Definition. The ability to work with multi choke points (of different values) at the same time, e.g. as in a VDQ implementation. The primary intended use of the multi choke fairness control message (MC-FCM).

9.3.34 norm_local_fair_rate: Calculated. The (NORMCOEF) normalized value of local_fair_rate. It is the value for advertised_fair_rate in SC-FCMs when the local station is more congested than downstream stations; and always the value for advertised_fair_rate in MC-FCMs.

9.3.35 norm_lp_fw_rate_congested: Calculated. The (NORMCOEF) normalized value of lp_fw_rate_congested.

9.3.36 NORMCOEF: Configured. The coefficient used for normalizing advertised fair rates to a common rate. The value for this constant is the product of AGECOEF, RATECOEF, and WEIGHT.

9.3.37 nr_xmit_rate: Calculated. A running byte count of all non reserved (non subclass-A0 marked) packets transmitted to the ringlet (from all add queues and transit queues) of the local station during the previous AGECOEF aging_intervals (including the current interval), and smoothed over the previous AGECOEF aging_intervals.

9.3.38 num_stations: Calculated. The number of stations that are known to be on the ringlet. Reported via rprIfNodesOnRing.

9.3.39 RAMPCOEF: Configured. The coefficient used for ramping up or down the allowed_rate_congested or local_fair_rate. The default value for this constant is 64. Values other than factors of 2 are not required to be supported.

9.3.40 RATECOEF: Configured. The coefficient used for normalizing advertised fair rates to a common link rate. The RATECOEF is used to ensure that the control value does not overflow 16-bits. The value to use for each supported link speed is shown in Table 9.2.

Table 9.2—Rate Coefficient

Link Speed	Rate Coefficient
Up to and including 2.5 Gbps	1
10 Gbps	4
40 Gbps	16

9.3.41 rcvd_fair_rate: Calculated. The normalized rate (in bytes) received in an FCM, giving the fair number of bytes that can be sent during the next aging_interval.

9.3.42 reserved_rate: Calculated. The total class A traffic that is reserved (i.e. the total of all sub class A0 traffic) on the outgoing link for all stations, including the local station, specified as the number of bytes per AGECOEF aging_intervals.

9.3.43 single choke: Definition. The ability to work with only one choke point at one time. The primary intended use of the single choke fairness control message (SC-FCM).

9.3.44 ~~add_rate~~**TTL_to_congestion:** Calculated. The ~~actual amount~~ number of ~~fairness eligible traffic sourced by~~ hops to the ~~node for the ringlet~~congestion point.

~~a) add_rate_congestion: The amount of fairness eligible traffic sourced by the node for the ringlet, and destined beyond the congestion point.~~

~~b) advertisement interval: The time interval at which Type A fairness messages are sent.~~

~~c) allow_rate_congestion: The maximum amount of fairness eligible traffic that the node is allowed to source for destinations beyond the congestion point.~~

~~d) allowed rate: The rate at which a node is allowed to transmit fairness eligible traffic on a given ringlet.~~

~~e) available_bandwidth: This is the (link speed - reserved bandwidth).~~

~~f) decay_interval: The decay interval is a timer that is used to perform fairness specific tasks every time it expires. It is defined in microseconds and defaults to 100 usec. Every time the decay interval timer expires, the MAC fairness control unit performs the following tasks:~~

1) ~~Set allow_rate_congestion with the weighted value in the received fairness message if the value is not null, otherwise ramp it up.~~
2) ~~Set the fill rate for the token bucket for fairness eligible traffic with the value corresponding to the allowed_rate_congestion (See 6.X).~~
3) ~~Determine if the node is congested or not.~~
4) ~~Low pass filtering operations are performed for add_rate, forward_rate, add_rate_congestion, and forward_rate_congestion.~~
5) ~~add_rate, add_rate_congestion, forward_rate, and forward_rate_congestion are decremented (decayed).~~

g) ~~fair rate: The fair rate is a normalized byte count measured over a decay interval.~~
h) ~~fairness eligible traffic: Traffic whose access to the ring is controlled based on the fairness algorithm. It consists of EIR Class B and Class C traffic only.~~
i) ~~forward_rate: The amount of fairness eligible traffic that transits through the station.~~
j) ~~forward_rate_congestion: The amount of fairness eligible traffic that transits through the station and that goes beyond the point of congestion.~~
k) ~~line_rate: The speed of the link expressed in bytes per decay interval.~~
l) ~~max_rate: The maximum possible transmit rate of fairness eligible traffic for the node.~~

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

More detail is probably needed for the definition of ramp up.

m) ~~ramp_up: Increasing the allowed rate at a station decay interval.~~
n) ~~reserved_rate: The reserved bandwidth at a node. This includes the total outgoing Class A0 traffic from the node. It is the sum of the transit Class A0 traffic and the Class A0 traffic sourced by the node.~~
o) ~~rcvd_advertised_rate: The control value received in a Type A message.~~
p) ~~STQ_HI_THRESH: A high threshold for the secondary transit queue in dual queue MAC.~~
q) ~~STQ_LO_THRESH: A low threshold for the secondary transit queue in a dual queue MAC.~~

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

Clarify which rates have a corresponding low pass filtered value that is maintained. Also clarify that the advertised rates are the low pass filtered ones. The low pass filtering options should be defined in the fairness clause. Contributions are requested for this.

9.3.45 unreserved_rate: Calculated. The rate available for the outbound link that is not reserved, specified as the number of bytes per AGECOEF aging_intervals.

9.3.46 WEIGHT: Configured. The local weight by which allowed_rate, allowed_rate_congested, and local_fair_rate are calculated. The default value for this constant is 1.

9.4 MAC fairness operation

> ***Editors' Notes:*** *To be removed prior to final publication.*
>
> *The following summary describes the generation of fairness messages. Make sure that the document is consistent with the following description.*
> *1) When a node is congested.*
> *How to generate Type As?*
> *If the node is more congested than the node from which*
> *it received the Type A message, then the local node advertises*
> *its fair rate (with its SA). Otherwise, the downstream node's*
> *value is passed without modifying the SA.*
> *How to generate Type Bs?*
> *Local node advertises its fair rate using its own SA and with*
> *a TTL of 255.*
> *2) When a node is not congested.*
> *How to generate Type As?*
> *If the following condition is true:*
> *(forward_rate_congestion > allow_rate_congestion)*
> *the downstream node's value is passed without*
> *modifying the SA. Otherwise, a NULL rate is sent in the Type A's.*
> *How to generate Type Bs?*
> *Local node advertises its fair rate using its own SA and with*
> *a TTL of 255. It may use a NULL value for the fair rate.*
> *Clarify that fairness messages are sent only at fairness intervals.*
> *A figure to be added describing operation of Type A's and Type B's.*

~~The RPR-FA is a local fairness mechanism that enforces fairness among the stations on the ring. It controls the amount of fairness eligible traffic (i.e. EIR Class B and Class C traffic) that the MAC is allowed to insert or each ringlet. RPR-FA is implemented within a control entity called the Fairness Control Unit (FCU). The MAC utilizes the topology information including the ringlet selection preference passed by the MAC client to perform fairness and policing functions.~~

The fairness algorithm implemented within the FCU consists of the following functions:

a) Determining when the congestion threshold is crossed and when the congestion has subsided;
b) Determining the fair rate for advertisement;
c) Determining the station's allowed rate;
d) Sourcing and consuming fairness control messages;
e) Communicating the allowed rate to the data path shapers for controlling access to the medium~~.~~;
f) Providing the information contained in MC-FCMs to the client.

Each station is assigned a weight, which allows the user to allocate more ring bandwidth to certain stations as compared with other stations. This is referred to as the *weighted fairness* property of RPR-FA.

In RPR-FA, a ~~node~~ station advertises a fair rate ~~(i.e. add_rate)~~ to upstream ~~nodes~~ stations via the ringlet opposite ~~ring~~of the ringlet upon which the algorithm is running. The fair rate ~~counter~~ is run through a low pass filter ~~function and divided~~ function, with the result known as the local_fair_rate. The local_fair_rate is normalized by ~~a weighting function (i.e~~ the local station ~~weight~~weight (WEIGHT), the aging coefficient (AGECOEF), and the rate coefficient (RATECOEF) to yield the norm_local_fair_rate and the advertised_fair_rate. The low-pass filter stabilizes the feedback, ~~and~~ the division by ~~weight~~ WEIGHT normalizes the transmitted value to a weight of 1.0, the division by AGECOEF normalizes the transmitted value to one aging_interval, and the division by RATECOEF normalizes the transmitted value to a link speed of 2.5 Gbps. When the upstream stations receive an advertised fair rate, they will adjust their transmit rates for fairness eligible traffic so as not to exceed the advertised value (adjusted by their respective weights).

Propagation of the advertised value to other ~~nodes~~ stations on the ring is done using fairness control messages. The format of the fairness control messages is described in 9.12 . There are 2 types of fairness control messages—~~Type A~~ Single Choke and ~~Type B~~Multi Choke.

Single Choke messages are propagated hop-by-hop around the opposite ringlet and are processed by the FCU. They are sent every advertisement_interval.

A wrapped ring shall be treated as a folded ring from the point of view of SC-FCMs. A station with a wrapped attachment point receiving a SC-FCM shall change the ringlet_id and wrap the SC-FCM. SC-FCMs are stripped only when received with SA equal to the local MAC address and with the correct ringlet_id.

Single Choke fairness control messages contain the SA of the most congested station. If a station experiences congestion, it will include a non-FULL_RATE value for the fair rate in its Single Choke fairness control messages. A station that receives a Single Choke message with a SA of its MAC address shall treat the message as having a control value of FULL_RATE regardless of the actual control value.

A station that in the congested state shall advertise the minimum of its local_fair_rate and the last received fair rate (known as the rcvd_fair_rate) in its Single Choke message.

~~Type~~ A ~~messages are propagated hop-by-hop around the opposite ringlet and are processed by the FCU. They are sent every advertisement interval. They contain the SA of the most congested node. If a node experiences congestion, it will include a non-NULL value for the fair rate in its Type A fairness messages. A node that receives an advertised value in a Type A message, and~~ station that is ~~also in the congested state, shall advertise the minimum of its normalized low pass filtered fair rate and the received fair rate in its Type A message. A node that is~~ not congested and that receives a ~~Type A~~ Single Choke message containing a non-~~NULL rcvd_advertised_rate~~ FULL_RATE rcvd_fair_rate shall either propagate the ~~rcvd_advertised_rate~~ rcvd_fair_rate to its upstream neighbor (leaving the SA the same), or it will send a value of ~~NULL~~ FULL_RATE and set the SA to its own. Which of these is sent is determined by the following. If the low pass filtered ~~forward_rate~~ fw_rate_congested is less than the ~~allow_rate_congestion~~ allowed_rate_congested divided by the local station weight, then a ~~NULL~~ FULL_RATE value is propagated to the upstream neighbor instead of the ~~rcvd_advertised_rate~~rcvd_fair_rate. Otherwise, there cannot be an upstream ~~node~~ station that is the cause of congestion. ~~-~~Thus there is no need to propagate the ~~Type A~~ Single Choke fairness control message indicating congestion upstream of this ~~node.~~ station.

The value of the rate that is allowed ~~rate~~ for the fairness eligible traffic (allowed_rate_congested) that can be added by the ~~node~~ station is derived from the received advertised rate in the ~~Type A~~ Single Choke messages and the local station ~~weight~~normalization factor (NORMCOEF). If the number of hops to the destination of ~~the~~ a data packet from the MAC client is less than the number of hops to the ~~node~~ station that generated the ~~Type A~~ Single Choke message, then the MAC client can take advantage of the available bandwidth on the ~~ring~~ringlet; otherwise, if the destination of ~~the~~ a data packet would allow the packet to go beyond the ~~node~~ station that generated the fairness control message, the client will at least receive its fair share of the bandwidth from the most congested span that it contends for~~. The fair rate is equal to the allowed rate adjusted by the local station weight~~.

~~Type B~~ Multi Choke messages are broadcast on the ringlet opposite to the ringlet upon which the fairness algorithm is running and contain the SA of the ~~node~~ station that originated the message. They are sent every 10 ~~advertisement intervals~~advertisement_intervals. If a ~~node~~ station experiences congestion, it will include a non-~~NULL~~ FULL_RATE value for the fair rate in its ~~Type B~~ Multi Choke fairness control messages; otherwise it will include a FULL_RATE value for the fair rate. ~~Type B~~ Multi Choke messages are not required to be processed by the FCU, but the information is passed to the MAC client by the FCU and is used by the MAC client as described below.

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

What do Type B messages contain if the node is not congested? According to the description in the previous editors' note (which was agreed to at the last comment resolution session) the rate may be a "fair rate" or it may be NULL. Since this seems to be a contentious issue, I have left it open.

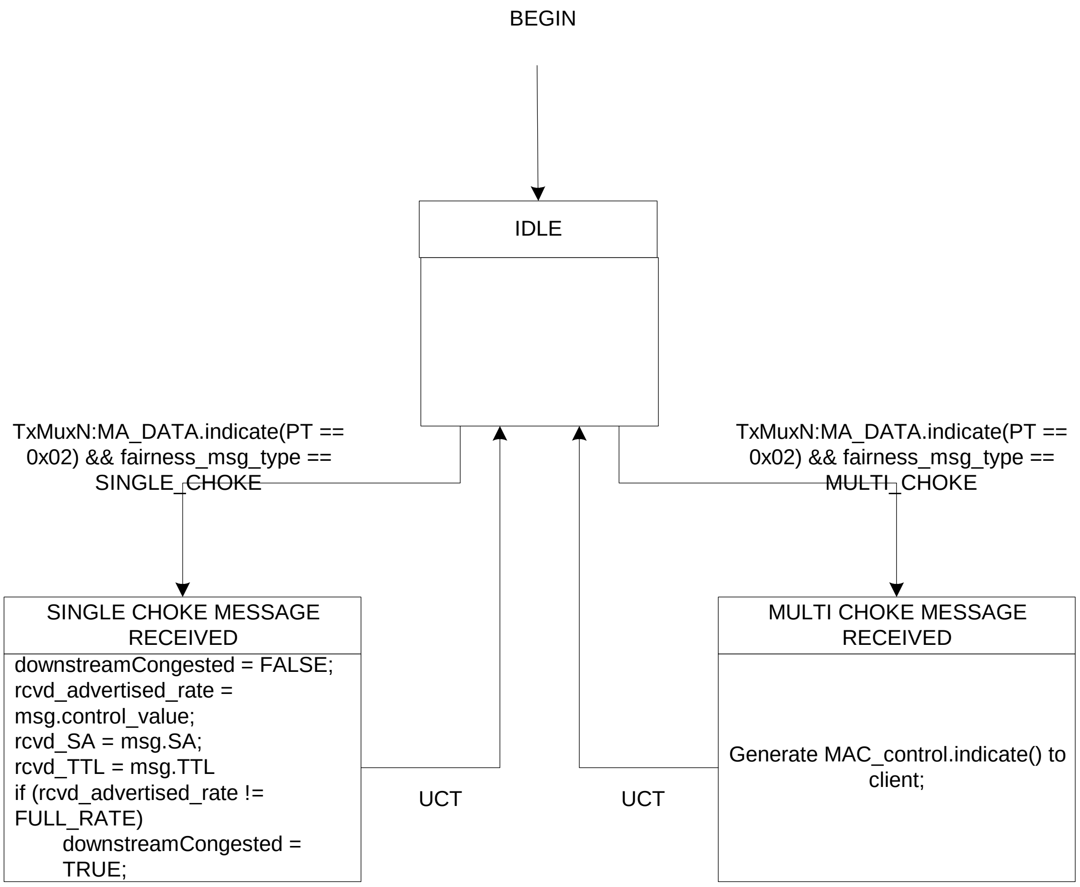
NOTE—The client can also take advantage of Virtual Destination Queueing (VDQ) by utilizing the multi-choke concept of RPR-FA. VDQ combined with RPR-FA can increase ring utilization. The multi-choke concept deals with the case where a ~~node~~ station wants to send traffic to a destination that is closer than a congested link. As an example, consider the case where ~~node~~ station 1 wants to send traffic to ~~node~~ station 2, and the link between ~~nodes~~ stations 2 and 3 is congested. RPR-FA will allow ~~node~~ station 1 to send as much traffic as it wants to ~~node~~ station 2, and will only limit traffic to ~~nodes~~ stations beyond the congested link to the fair rate. In a multi-choke implementation of the RPR-FA, each client will track advertised fair rates for congested ~~nodes~~stations. A ~~node~~ station is allowed to send unlimited traffic to any ~~node~~ station between itself and the first congested ~~node~~ station (choke point). It can send traffic to ~~nodes~~ stations between the first and second choke point based on the first choke point's advertised fair rate. In general, a ~~node~~ station can send traffic to a particular destination if it has satisfied the fair rate conditions for all choke points between itself and the destination. The maximum possible number of choke points is equal to the number of ~~nodes~~ stations on the ring.

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

The following state diagrams are intended to provide additional information to help in the understanding and completion of the state tables that follow them. These need to be reconciled with the tables and the text.

Part 1 of 7.

Receive Fairness Message

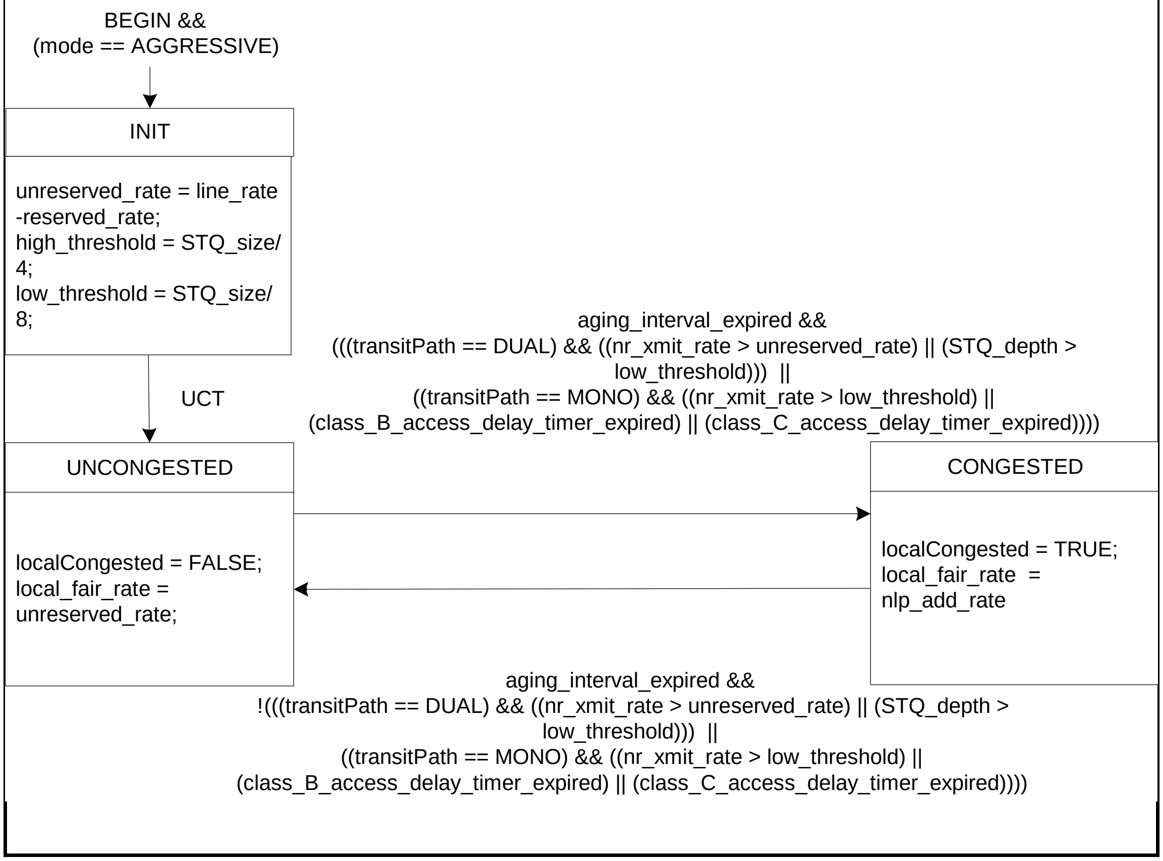

BEGIN
IDLE
TxMuxN:MA_DATA.indicate(PT == 0x02) && fairness_msg_type == SINGLE_CHOKE
TxMuxN:MA_DATA.indicate(PT == 0x02) && fairness_msg_type == MULTI_CHOKE
SINGLE CHOKE MESSAGE RECEIVED
downstreamCongested = FALSE;
rcvd_advertised_rate = msg.control_value;
rcvd_SA = msg.SA;
rcvd_TTL = msg.TTL
if (rcvd_advertised_rate != FULL_RATE)
downstreamCongested = TRUE;
UCT
UCT
MULTI CHOKE MESSAGE RECEIVED
Generate MAC_control.indicate() to client;

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

Continuation of editor's note.

Part 2 of 7.

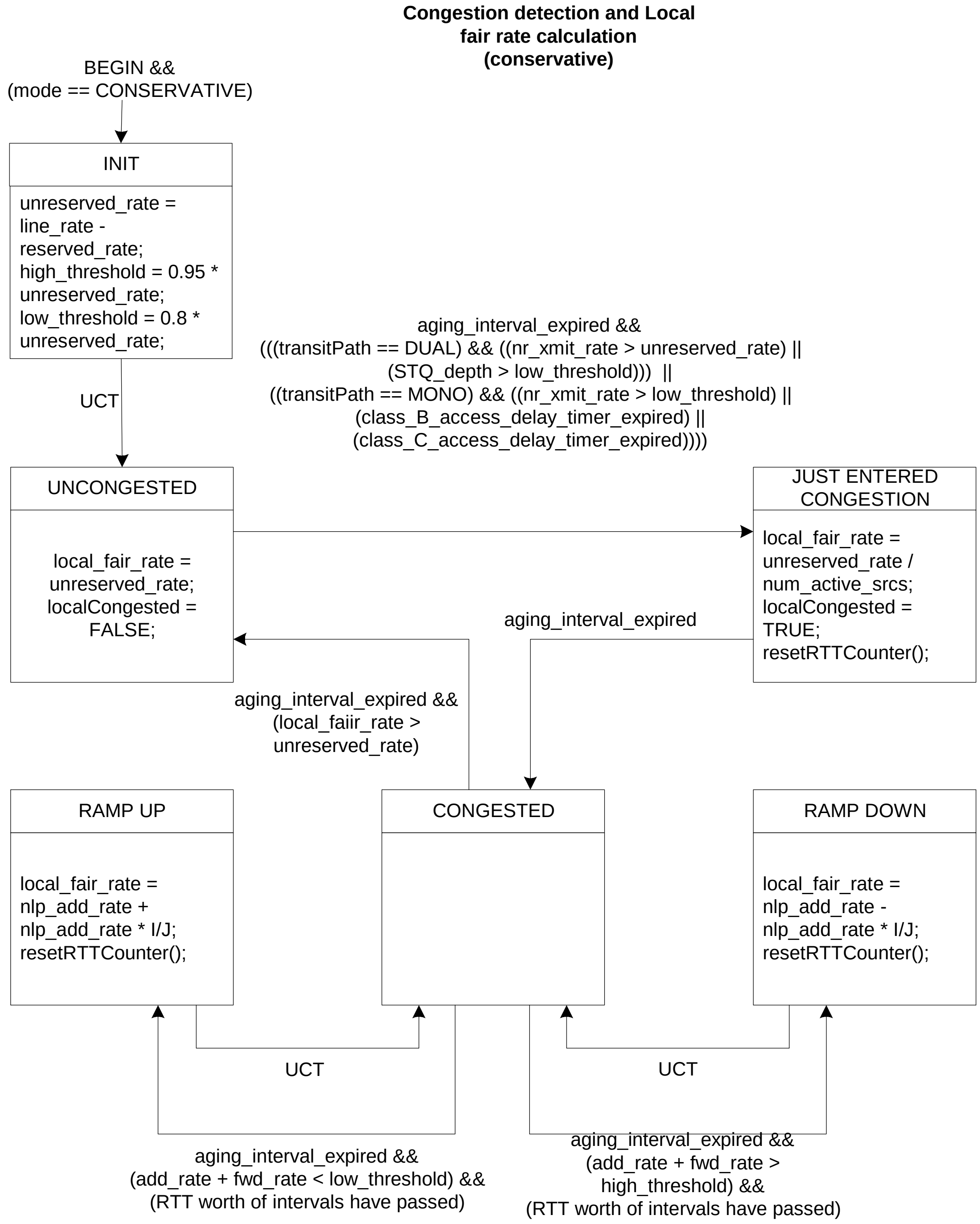
Congestion detection and local fair rate calculation (aggressive)


BEGIN &&
(mode == AGGRESSIVE)
INIT
unreserved_rate = line_rate -reserved_rate;
high_threshold = STQ_size/ 4;
low_threshold = STQ_size/ 8;
UCT
aging_interval_expired &&
(((transitPath == DUAL) && ((nr_xmit_rate > unreserved_rate) || (STQ_depth > low_threshold))) ||
((transitPath == MONO) && ((nr_xmit_rate > low_threshold) ||
(class_B_access_delay_timer_expired) || (class_C_access_delay_timer_expired))))
UNCONGESTED
localCongested = FALSE;
local_fair_rate = unreserved_rate;
CONGESTED
localCongested = TRUE;
local_fair_rate = nlp_add_rate
aging_interval_expired &&
!(((transitPath == DUAL) && ((nr_xmit_rate > unreserved_rate) || (STQ_depth > low_threshold))) ||
((transitPath == MONO) && ((nr_xmit_rate > low_threshold) ||
(class_B_access_delay_timer_expired) || (class_C_access_delay_timer_expired))))

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

Continuation of editor's note.

Part 3 of 7.

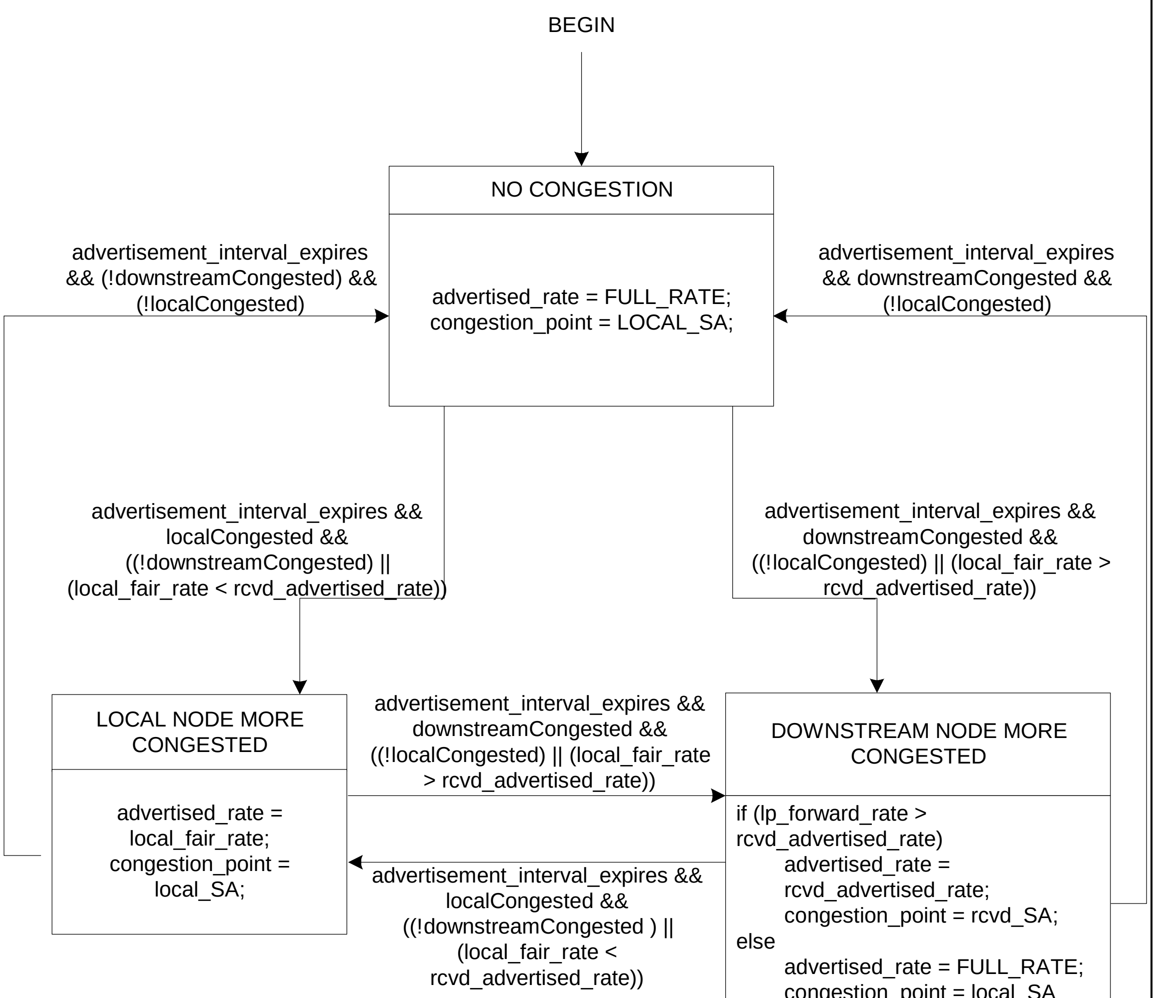

Congestion detection and Local fair rate calculation (conservative)
BEGIN && (mode == CONSERVATIVE)
INIT
unreserved_rate = line_rate - reserved_rate; high_threshold = 0.95 * unreserved_rate; low_threshold = 0.8 * unreserved_rate;
UCT
UNCONGESTED
local_fair_rate = unreserved_rate; localCongested = FALSE;
aging_interval_expired && (((transitPath == DUAL) && ((nr_xmit_rate > unreserved_rate) || (STQ_depth > low_threshold))) || ((transitPath == MONO) && ((nr_xmit_rate > low_threshold) || (class_B_access_delay_timer_expired) || (class_C_access_delay_timer_expired))))
JUST ENTERED CONGESTION
local_fair_rate = unreserved_rate / num_active_srcs; localCongested = TRUE; resetRTTCounter();
aging_interval_expired
aging_interval_expired && (local_faiir_rate > unreserved_rate)
RAMP UP
local_fair_rate = nlp_add_rate + nlp_add_rate * I/J; resetRTTCounter();
CONGESTED
RAMP DOWN
local_fair_rate = nlp_add_rate - nlp_add_rate * I/J; resetRTTCounter();
UCT
UCT
aging_interval_expired && (add_rate + fwd_rate < low_threshold) && (RTT worth of intervals have passed)
aging_interval_expired && (add_rate + fwd_rate > high_threshold) && (RTT worth of intervals have passed)

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

Continuation of editor's note.

Part 4 of 7.

Generation of single choke message

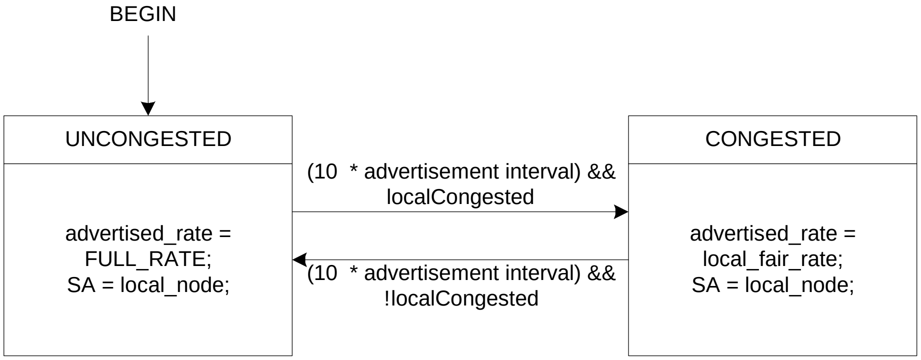

BEGIN
NO CONGESTION
advertised_rate = FULL_RATE;
congestion_point = LOCAL_SA;
advertisement_interval_expires && (!downstreamCongested) && (!localCongested)
advertisement_interval_expires && downstreamCongested && (!localCongested)
advertisement_interval_expires && localCongested && ((!downstreamCongested) || (local_fair_rate < rcvd_advertised_rate))
advertisement_interval_expires && downstreamCongested && ((!localCongested) || (local_fair_rate > rcvd_advertised_rate))
LOCAL NODE MORE CONGESTED
advertised_rate = local_fair_rate;
congestion_point = local_SA;
advertisement_interval_expires && downstreamCongested && ((!localCongested) || (local_fair_rate > rcvd_advertised_rate))
advertisement_interval_expires && localCongested && ((!downstreamCongested) || (local_fair_rate < rcvd_advertised_rate))
DOWNSTREAM NODE MORE CONGESTED
if (lp_forward_rate > rcvd_advertised_rate)
advertised_rate = rcvd_advertised_rate;
congestion_point = rcvd_SA;
else
advertised_rate = FULL_RATE;
congestion_point = local_SA

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

Continuation of editor's note.

Part 5 of 7.

Generation of multi choke message

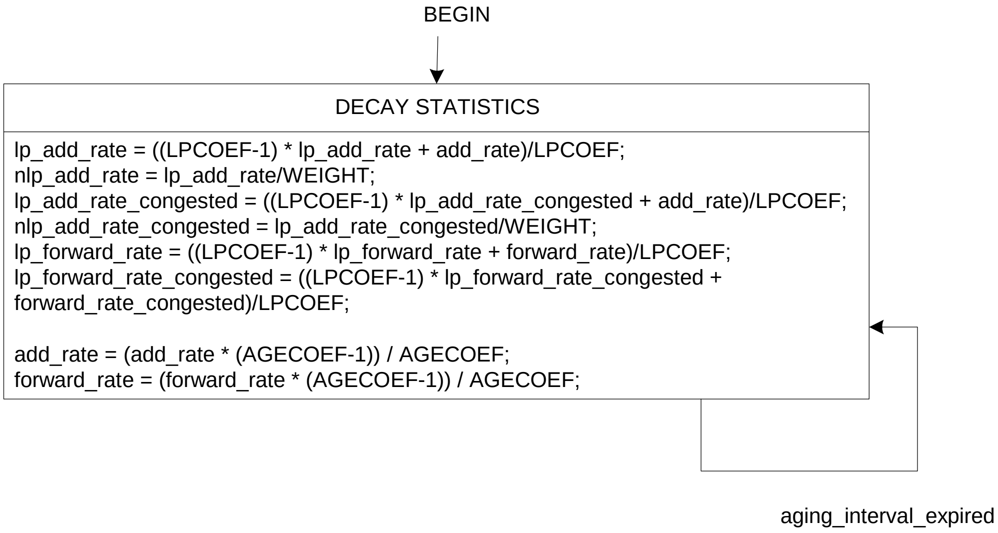

BEGIN
UNCONGESTED
advertised_rate = FULL_RATE; SA = local_node;
CONGESTED
advertised_rate = local_fair_rate; SA = local_node;
(10 * advertisement interval) && localCongested
(10 * advertisement interval) && !localCongested

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

Continuation of editor's note.

Part 6 of 7.

Statistics Decay

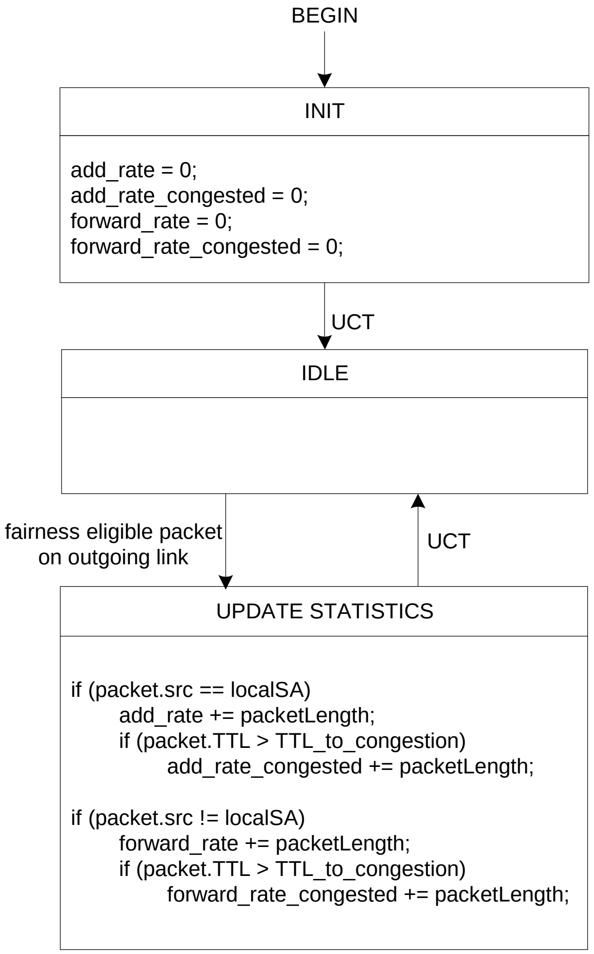

BEGIN
DECAY STATISTICS
lp_add_rate = ((LPCOEF-1) * lp_add_rate + add_rate)/LPCOEF;
nlp_add_rate = lp_add_rate/WEIGHT;
lp_add_rate_congested = ((LPCOEF-1) * lp_add_rate_congested + add_rate)/LPCOEF;
nlp_add_rate_congested = lp_add_rate_congested/WEIGHT;
lp_forward_rate = ((LPCOEF-1) * lp_forward_rate + forward_rate)/LPCOEF;
lp_forward_rate_congested = ((LPCOEF-1) * lp_forward_rate_congested + forward_rate_congested)/LPCOEF;
add_rate = (add_rate * (AGECOEF-1)) / AGECOEF;
forward_rate = (forward_rate * (AGECOEF-1)) / AGECOEF;
aging_interval_expired

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

Continuation of editor's note.

Part 7 of 7.

Statistics Update



```
add_rate = 0;
add_rate_congested = 0;
forward_rate = 0;
forward_rate_congested = 0;
```

```
if (packet.src == localSA)
    add_rate += packetLength;
    if (packet.TTL > TTL_to_congestion)
        add_rate_congested += packetLength;

if (packet.src != localSA)
    forward_rate += packetLength;
    if (packet.TTL > TTL_to_congestion)
        forward_rate_congested += packetLength;
```

9.4.1 FCM Processing

9.4.1.1 FCM Receive

For every aging interval, all variables used to determine congestion condition, congestion location, and rate control are recomputed. Those values that depend upon values received in SC-FCMs use the most recently received SC-FCM. The flow chart and state machine for processing received FCMs are as follows:

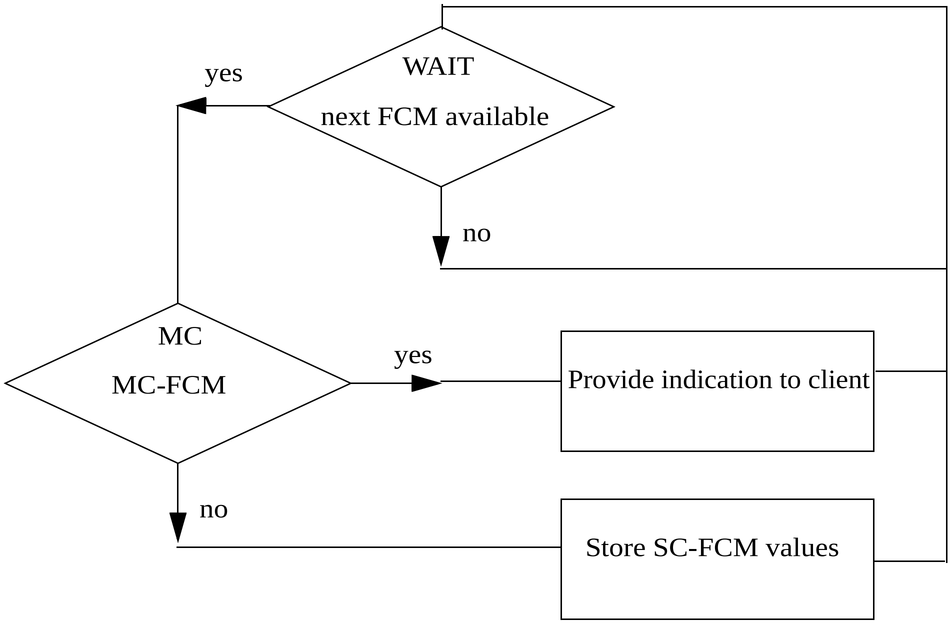


Figure 9—1— Fairness Control Message Reception

Table 9.3—FCM Reception States

Last state		**Row**	**Next state**	
state	**condition**		**action**	**state**
MC	MC-FCM	9.0.1	MA_CONTROL.indication(msg);	WAIT

	—	9.0.2	rcvd_fair_rate = msg.control_value;
			rcvd_SA = msg.SA;
			rcvd_TTL = msg.TTL;	

WAIT	next FCM available	9.0.3	—	MC
	—	9.0.4	—	WAIT

Row 9.0.1: The FCM is a MC-FCM. Make its relevant values available to the client.
Row 9.0.2: The FCM is a SC-FCM. Store its relevant values for use after the next aging interval expires.

Row 9.0.3: When a new FCM is received, start processing it.
Row 9.0.4: Wait for another FCM.

9.4.1.2 Local fair rate calculation

Using the most recently received SC-FCM, the local fair rate is recomputed every aging interval.

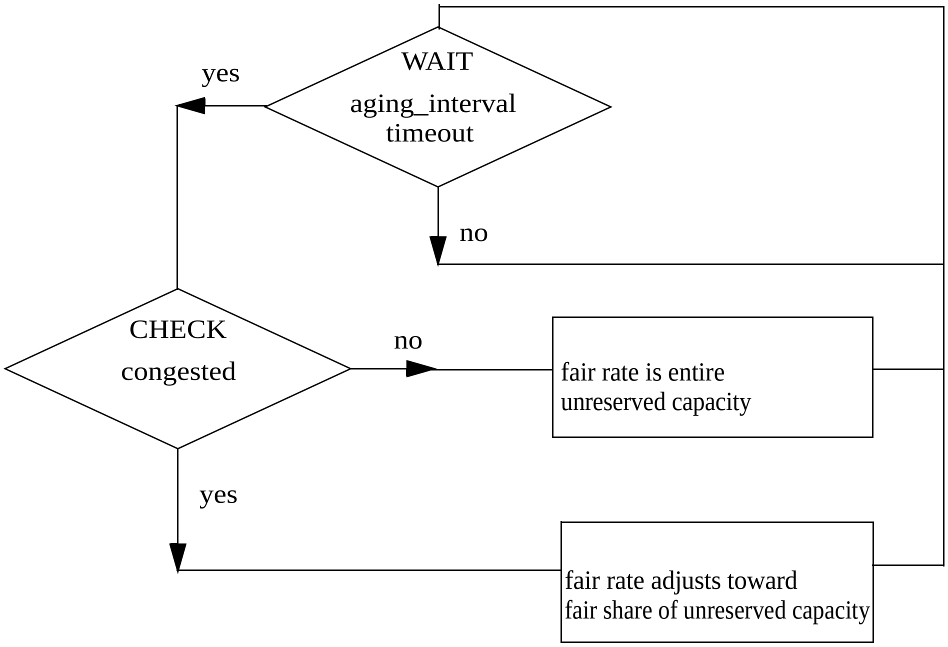


Figure 9—2—Local Fair Rate Calculation

Table 9.4—Local Fair Rate Calculation States

Last state		Row	Next state	
state	**condition**		**action**	**state**
CHECK	!congested	9.0.1	local_fair_rate = unreserved_rate;	WAIT
	aggressive	9.0.2	local_fair_rate = lp_add_rate;	
	just_entered_congestion	9.0.3	set local_fair_rate to weight adjusted equal share of unreserved_rate	
	FE transmit rate below low_threshold	9.0.4	ramp up local_fair_rate	
	FE transmit rate above high_threshold	9.0.5	ramp down local_fair_rate	
	—	9.0.6	—	
WAIT	aging_interval timeout	9.0.7	recalculate the world;	CHECK
	—	9.0.8	—	WAIT

Row 9.0.1: Outgoing link is not congested. Local station can try to use entire (unreserved) capacity.
Row 9.0.2: Aggressively try to use current add rate.
Row 9.0.3: When first becoming congested, try to use an equal share of the unreserved capacity, adjusted for station weight.
Row 9.0.4: FE transmit rate is too low. Ramp up fair rate towards unreserved capacity.
Row 9.0.5: FE transmit rate is too high. Ramp down fair rate towards 0.
Row 9.0.6: FE transmit rate is not too low and not too high. Don't change amount of outgoing link currently trying to use.
Row 9.0.7: When the next aging_interval expires, recalculate all aging_interval dependent variables, except for local_fair_rate and norm_local_fair_rate.

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

Should 9.8.3 be referenced, or should the calculations/formulas be stated in this section? Also, is it helpful to describe the calculation of local_fair_rate as state machine instead of just a formula?

Row 9.0.8: Wait for the aging_interval to expire.

9.4.2 FCM Transmit

9.4.2.1 SC-FCM Transmit

For every advertisement_interval, the advertised_fair_rate and congested SA are recomputed (as described in 9.13.4) and advertised in a SC-FCM.

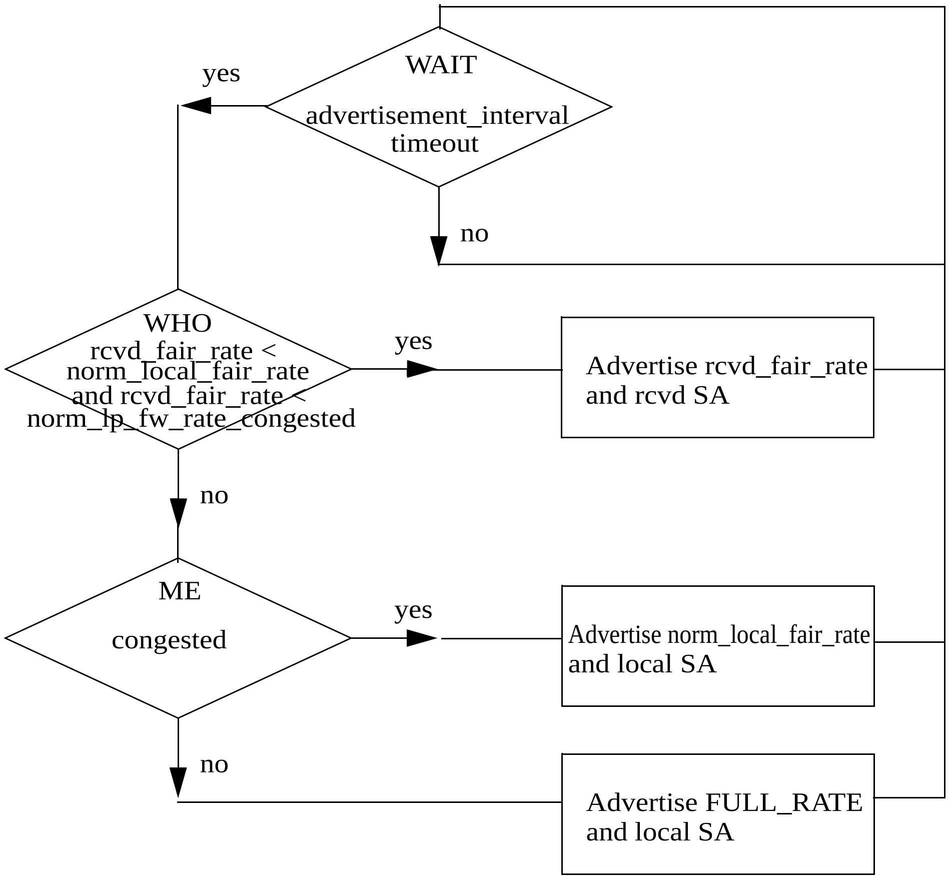


Figure 9—3—Single Choke Fairness Control Message Advertisement

Table 9.5—SC-FCM Advertisement States

Last state		Row	Next state	
state	**condition**		**action**	**state**

WHO	rcvd_fair_rate < norm_local_fair_rate &&
	rcvd_fair_rate < norm_lp_fw_rate_congested
		9.0.1	advertised_fair_rate = rcvd_fair_rate;
			congestion_point = rcvd_SA;
				WAIT

	—	9.0.2	—	ME

ME	congested	9.0.3	advertised_fair_rate = norm_local_fair_rate;
			congestion_point = local_SA;
				WAIT

	—	9.0.4	advertised_fair_rate = FULL_RATE;
			congestion_point = local_SA;	

WAIT	advertisement_interval timeout	9.0.5	—	WHO
	—	9.0.6	—	WAIT

Row 9.0.1: A downstream station is more congested than the local station.

Row 9.0.2: No downstream station is more congested than the local station

Row 9.0.3: The local station is congested.

Row 9.0.4: The local station (and therefore all stations) is not congested.

Row 9.0.5: When the next advertisement_interval expires, transmit another SC-FCM.

Row 9.0.6: Wait for the advertisement_interval to expire.

9.4.2.2 MC-FCM Transmit

For every 10 advertisement_intervals, the local_fair_rate is advertised in a MC-FCM.

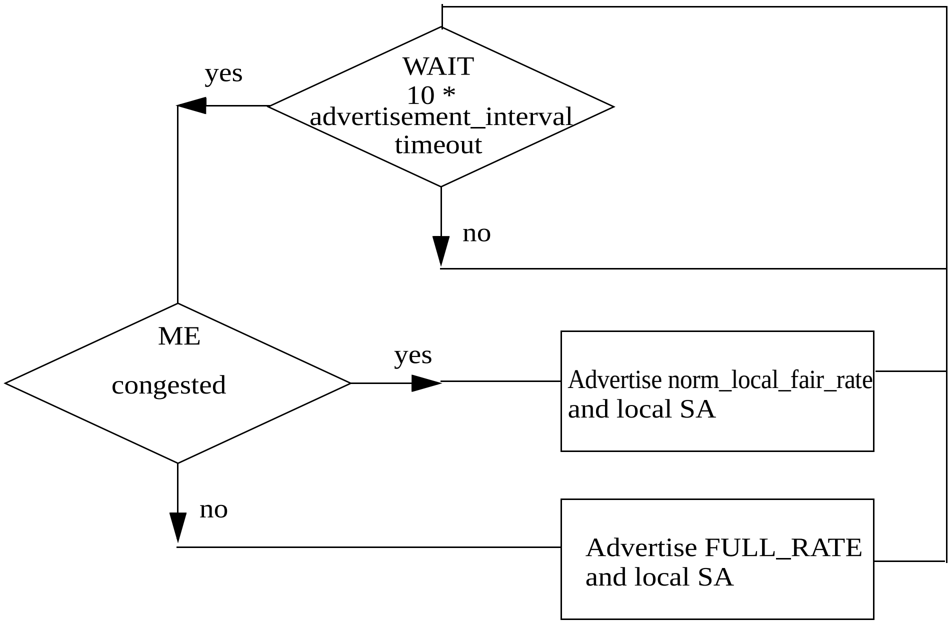


Figure 9—4—Multi Choke Fairness Control Message Advertisement

Table 9.6—MC-FCM Advertisement States

Last state		**Row**	**Next state**	
state	**condition**		**action**	**state**

ME	congested	9.0.1	advertised_fair_rate = norm_local_fair_rate;
			congestion_point = local_SA;
				WAIT

	—	9.0.2	advertised_fair_rate = FULL_RATE;
			congestion_point = local_SA;	

WAIT	10 * advertisement_interval timeout	9.0.3	—	ME
	—	9.0.4	—	WAIT

Row 9.0.1: The local station is congested.
Row 9.0.2: The local station is not congested.
Row 9.0.3: When 10 advertisement_intervals expire, transmit another MC-FCM.
Row 9.0.4: Wait for 10 advertisement_intervals to expire.

9.5 Congestion detection

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

In the text below, we need to have a better defintion for "configured thresholds" to detect coming out of congestion.

Congestion is declared by using the following ~~criteria depending on the number of transit queues used by the MAC~~criteria. The formula for determining congestion is specified in 9.13.3.4 .

~~If the MAC uses a mono transit queue design, congestion~~ Congestion is detected when any of the following conditions is true:

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

How does this handle CIR class B traffic? It is not accounted for in either reserved rate not in the add and forward rates. If is included in the reserved rate, then it prevents unused CIR Class B bandwidth from being reclaimed.
Also, we should probably define HI and LO link utilization thresholds and queue thresholds that we use for hysterisis for congestion detection and coming out of congestion. Contributions are invited for this.

~~a) The rate of outgoing non-reserved traffic passes a configured congestion threshold on the output link; i.e. if the sum of Class A1, Class B, and Class C traffic on the outgoing link is more than (line_rate – reserved_rate). The outgoing link rate is measured using a byte counter which is passed through a low pass filter before the comparison. Further, the node will be considered to be congested until its outgoing link utilization falls below a configured threshold.~~

a) The rate of outgoing non-reserved traffic (lp_nr_xmit_rate) is more than the unreserved rate (i.e. LINK_RATE – reserved_rate), or the low_threshold, if specified as a rate.

b) The access delay timer for Class B add packets expires.

c) The ~~access delay timer for Class B add packets or the~~ access delay timer for Class C add packets expires.

~~If the MAC uses a dual transit queue design, congestion is detected when any of the following conditions is true:~~

~~a) The rate of outgoing non-reserved traffic passes a configured congestion threshold on the output link; i.e. if the sum of Class A1, Class B, and Class C traffic on the outgoing link is more than (line_rate – reserved_rate). The outgoing link rate is measured using a byte counter which is passed through a low pass filter before the comparison. Further, the node will be considered to be congested until its outgoing link utilization falls below a configured threshold.~~

~~b) The depth of the STQ reaches a congestion threshold.~~

c) The depth of the STQ exceeds the low_threshold congestion depth threshold, or the rate of non-A0 traffic exceeds the low_threshold congestion rate threshold.

Conditions b) and c) are optional for dual-queue MACs. Condition c) is not applicable for mono-queue MACs.

The above metrics are maintained independently for each ringlet. Any of them could indicate the onset of congestion for the ringlet. The absence of all that are maintained indicates the lack of congestion.

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

Adding (optional) hysteresis to each of a - d need to be added.

The access delay timers are maintained by the MAC for each ringlet. There are separate access delay timers for Class B and Class C traffic. The timer corresponding to the class of packet is started at the time the packet becomes the head-of-line packet in the MAC. The timer is reset when the MAC successfully transmits that packet. If the MAC is unable to transmit the packet before the timer expires, the MAC enters a congested state. The access delay timers will have a range of 1-10 msec, with a default value of 5 msec.

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

Is there any justification for the above numbers? It should probably depend at least on the link speed and the RTT.

For Class A traffic there is no access delay timer. The access delay timer expiring for Class A traffic means there is something wrong with respect to provisioning. Perhaps OAM&P should look into providing an alarm for this condition. However, this issue doesn't concern fairness.

~~For Class B and Class C traffic, the access delay timer will have a range of 1-10 msec with a default value of 5 msec.~~

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

Is there any justification for the above numbers? It should probably depend at least on the link speed and the RTT.

For Class A traffic there is no access delay timer. The access delay timer expiring for Class A traffic means there is something wrong with respect to provisioning. Perhaps OAM&P should look into providing an alarn for this condition. However, this issue doesn't concern fairness.

~~9.6 Interoperability between mono and dual queue MACs~~

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

There is no new information in this subclause. Suggest deleting it. Unless we want to say something about shaping the STQ in a dual queue transit path design to enable interoperability between mono and dual transit queue MACs.

~~If a ring consists of mixed RPR MACs (mono and dual transit queue nodes), the fairness scheme will need to interact without disadvantaging any other node on the ring. The actions required for the interaction between mono and dual transit queue nodes is the same as the interaction between similar nodes. Upon receiving a Type A fairness message the RPR MAC shall reduce its allowed rate to the fair rate received in the fairness message multiplied by the local weight, and then forward the message upstream with the minimum of its own advertised rate and the received fair rate.~~

~~The RPR MAC should rate-shape its transmit traffic using the dynamic traffic shaping algorithm described in XX—Dynamic traffic shaping.~~

~~9.7 Threshold settings for dual transit queue RPR MACs~~

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

Since this clause deals with the onset of congestion, the actual threshold settings probably belong in the clause on congestion detection. The rationale, etc. should probably be moved to Annex I: Implementation Guidelines.

~~Two registers, STQ_HI_THRESH and STQ_LO_THRESH, are used to specify congestion thresholds within a large STQ. When the occupancy of the STQ crosses STQ_LO_THRESH, the node is considered to be in a congested state, and no more fairness eligible traffic is accepted from the MAC client for transmission on to the ringlet. If the occupancy of the STQ crosses STQ_HI_THRESH, it is almost full and the traffic from the STQ will be prioritized above all traffic from the MAC client until the occupancy of the STQ drops below STQ_HI_THRESH.~~

When a station determines that it is a congestion point, it will send a fairness control message containing a non-FULL_RATE fair rate, based on a normalized value of it low pass filtered add rate.

9.7.1 Threshold settings

Three thresholds, full_threshold, high_threshold, and low_threshold, are used to specify congestion thresholds. For dual-queue MACs, they are based on occupancy of the STQ. For mono-queue MACs, they are based on the rate at which traffic is flowing through the MAC. The formulas for calculating these thresholds are provided in 9.13.1 . When the low_threshold is crossed, the station is considered to be approaching a congested state, and advertises a reduced fair rate. When the high_threshold is crossed, the station is considered to be in a congested state, and no more fairness eligible traffic is accepted from the MAC client for transmission on to the ringlet. For dual-queue MACs, if the occupancy of the STQ crosses full_threshold, it is almost full, and the traffic from the STQ will be prioritized above all other traffic until the occupancy of the STQ drops below full_threshold (by any amount).

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

The remainder of this section deals with rationale, and should probably be moved to Annex I: Implementation Guidelines.

The setting of ~~STQ_LO_THRESH~~ high_threshold depends on the following factors:

a) If set too low, it will trigger congestion reports too frequently which in turn will adversely affect the link utilization.

b) If set higher, the difference between the ~~STQ_LO_THRESH~~ high_threshold and ~~STQ_HI_THRESH~~ full_threshold is reduced. Because this reduces the time between congestion onset and the STQ filling up, it has the following effect:
1) For a given level of reclaimable Class A bandwidth (i.e. Class A1 bandwidth), the link distances must be shorter.
2) For a given link distance, a smaller level of reclaimable Class A traffic can be supported. (See Clause 6.x where this calculation is provided.)

The more severe consequences of a) vs. b) are the reason that a threshold of less than 50% is chosen. This recommendation applies only when a STQ of many MTUs is used to support high amounts of reclaimable Class A bandwidth.

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

As requested at the May meeting, Vasan Karighattam will have a contribution for an analytical method to compute the STQ size.

The primary transit queue needs to hold at least 2 MTUs.

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

As requested at the May meeting, Necdet Uzun and David James will have a justification for this threshold.

~~STQ_LO_THRESH should be set to about 25% of the total buffer available. STQ_HI_THRESH should be set to at least 1 MTU less than the total buffer size.~~

NOTE—high threshold should be set to about 25% of the total buffer available. full threshold should be set to at least 1 MTU less than the total buffer size. The recommendation for the setting of the threshold values is done based on delivering the best possible end-to-end delay for the low priority traffic without penalizing the Class A traffic. Lower values will result in higher end-to-end delays for low priority data packets. If either Class A, B, or C traffic is extremely bursty, then a lower threshold value should be considered. A ~~STQ_HI_THRESH~~ full threshold of at least 1 MTU less than the total buffer size is needed to ensure that the ~~node~~ station has sufficient buffer space for a packet that may arrive on the transit path. If the Class A traffic has a bursty ~~nature~~ nature, a more conservative (i.e. lower) value of ~~STQ_LO_THRESH~~ high threshold is recommended in order to avoid overflow of the STQ.

~~9.8 Traffic policing function~~

~~RPR-FA utilizes the allow_rate_congestion and TTL_to_congestion registers along with the add_rate_congestion counter to police the fairness eligible traffic that is added by the node to the ringlet. The allow_rate_congestion and TTL_to_congestion registers store the values of the fair rate and the distance to the choke point of the most recently received Type A fairness packet, respectively. The add_rate_congestion counter accumulates the number of bytes of fairness eligible traffic sent beyond the congestion point on the ringlet (i.e., only packets whose destination is further than TTL_to_congestion contribute to add_rate_congestion). If the add_rate_congestion exceeds allow_rate_congestion, backpressure signals are generated to the client.~~

~~9.9 Dynamic traffic shaping~~

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

This section should probably be combined with the previous one, into a subsection called Access Control.

~~RPR-FA limits the ring access rate for fairness eligible traffic destined beyond the congestion point to within allow_rate_congestion. Due to the dynamic nature of RPR-FA, allow_rate_congestion can vary widely from one decay interval to the next. When access traffic contains transmission bursts, its rate can vary significantly and is only limited by allow_rate_congestion. As a result, traffic on the ring can be bursty. To achieve better jitter performance, the ring access rate should be dynamically shaped to the allow_rate_congestion, which allows ring access rate conform to a fair rate as governed by RPR-FA. This reduces extra bursty transmission.~~

9.10 Interaction with data path

RPR-FA uses the add_rate, add_rate_congested, fw_rate, and fw_rate_congested values provided by the data path, as specified in Clause 6.

RPR-FA provides the add_rate_ok, add_rate_congested_ok, allowed_rate, allowed_rate_congested, and TTL_to_congestion values to the data path rate control function to police the fairness eligible traffic that is added by the station to the ringlet. The add_rate_ok and add_rate_ok_congested variables are used to enable or disable the sendC signal and to determine if the TTL_to_congestion value should be applied. The TTL_to_congestion variable stores the value of the distance to the present or more recent choke point. The allowed_rate_congested variable stores the value of the fair rate for fairness eligible traffic passing through the congestion_point, as given by the most recently received Single Choke fairness control message. The allowed_rate variable stores the value of the rate for all fairness eligible traffic. The data path maintains rate shapers to limit the add_rate (Sd) and add_rate_congested (Sc) values below the provided allowed_rate and allowed_rate_congested values, respectively, and to shape their output at or below the provided rates to smooth bursty transmissions.

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

This following paragraph belongs in clause 6.

The RPR-FA dynamic shaper consists of a token bucket. The token bucket has a default maximum depth of MTU bytes. The tokens are generated at a rate equal to the ~~allow_rate_congestion~~allowed_rate_congested. For every byte that is transmitted, the number of tokens is decremented by one. When a packet is waiting for access to the ring at the head of the queue, it will be granted ring access only if its rate conforms to the fair rate as computed by RPR-FA and there is at least one byte in the token bucket. It is therefore possible for the number of tokens to become negative after subtracting the number of bytes in the transmitted packet. The maximum number of tokens that can accumulate in the token bucket is equal to the size of an MTU.

~~9.11 RPR ring access operation~~

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

This whole section is kind of redundant with the clause on "RPR fairness overview". These should probably be merged.

~~The RPR-FA governs access to the ring. The RPR-FA only applies to fairness eligible traffic. Class A and "within CIR" Class B traffic does not follow RPR-FA rules and may be transmitted at any time as long as traffic shaping/policing allow it, STQ is not almost full and PTQ does not have a packet.~~

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

Once we define all the variables, we don't need the following paragraph.

~~The RPR-FA requires few counters/registers which control the traffic forwarded and sourced on the RPR ring. Those counters are add_rate (tracks the amount of Class C and EIR Class B traffic sourced on the ring), add_rate_congestion (tracks the amount of Class C and EIR Class B traffic sourced on the ring and destined beyond the congestion point) and forward_rate (amount of Class C and EIR Class B traffic transited through the station from the STQ), allow_rate_congestion (the current maximum Class C+EIR Class B transmit rate~~

~~for that node beyond the congestion point) and max_rate (the current maximum Class C+EIR Class B transmit rate for that node).~~

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

This paragraph belongs to the section on access control.

~~The traffic policer shall not allow add_rate_congestion and add_rate to exceed allow_rate_congestion and max_rate, respectively. It is accomplished by generating "not SendC" signal to the client when add_rate_congestion and add_rate exceed allow_rate_congestion and max_rate, respectively. (See 6.X for details on the SendC signal.)~~

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

The next few sections should probably be merged with the "MAC fairness operation section".

~~When a node receives a Type A fairness message that contains a NULL value, it shall increase its allow_rate_congestion every decay interval. The maximum value for allow_rate_congestion is max_rate. The max_rate is a per node parameter that limits the maximum amount of fairness eligible traffic that a node can send.~~

~~When a node sees congestion it starts to advertise a normalized add_rate value to upstream nodes. The normalized fair rate (nlp_add_rate) is obtained by passing add_rate through a low pass filter and then dividing by its weight. In this way, the fair rates passed on the links are always normalized to a weight of 1.0. Congestion detection is described in 9.5~~

~~When a node determines that it is a congestion point, it will send a fairness message containing a non-NULL fair rate. A node that receives a fairness message with a non-NULL fair rate (rcvd_advertised_rate) will set its allow_rate_congestion and TTL_to_congestion to the rcvd_advertised_rate value multiplied by its weight and 256 - received TTL value, respectively. This allows a node with a weight of N to utilize N times as much bandwidth as a node with a weight of 1.0. If the source of the rcvd_advertised_rate is the same node that received it then the rcvd_advertised_rate should be disregarded. When comparing the rcvd_advertised_rate source address the ring identifier of the fairness packet must match the receiver's ring identifier in order to qualify as a valid compare. The exception is if the receive node is in the wrap state in which case the fairness packet's ring identifier is ignored.~~

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

Per a comment on D0.2, the reference to convergence time of 100 msec for rings of several hundred miles was removed. The same comment also requested a formula for the convergence time, for which contributions are solicited.

9.12 Fairness control messages

9.12.1 Generation of fairness control messages

~~Type A~~ Single Choke and ~~Type B~~ Multi Choke fairness control messages shall be sent periodically to propagate fair rate information to upstream stations. ~~Type A~~ Single Choke messages are sent every ~~advertisement interval~~advertisement_interval. The recommended ~~advertisement interval~~ advertisement_interval is between the transmission delay for a single MTU and one-half of the ~~decay interval~~aging_interval. ~~Type B~~ Multi Choke messages are sent every 10 ~~advertisement intervals~~advertisement_intervals.

~~9.12.2 Receipt of fairness messages~~

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

Since this is covered by the MAC fairness operation clause, we can probably remove this (or move the appropriate text here.

9.12.3 Impact of lost fairness control messages

The ~~allow_rate_congestion~~ allowed_rate_congested maintains its current value until the next ~~Type A~~ Single Choke fairness control message is received. Therefore, the loss of a ~~Type A~~ Single Choke fairness control message would result in one of the following. If the lost message contained a ~~NULL value~~new value of FULL_RATE, it would prevent the ~~node~~ station from ramping up its allowed rate for the ringlet. If the value contained in the lost message was a new non-~~NULL~~FULL_RATE value, the MAC will not be able to react by setting its allowed rate to the received advertised rate, which could have been higher or lower than the current allowed rate. If the lost message did not contain a new value from the previous advertisement, there will be no effect on the fairness algorithm.

Refer to Clause 10 for protection implications of not receiving consecutive Single Choke fairness control messages.

The ~~Type B~~ Multi Choke messages are not processed by the MAC, and therefore loss of ~~Type B~~ Multi Choke messages doesn't impact behavior of the MAC.

~~Refer to Clause XX for protection implications of not receiving consecutive Type A fairness messages.~~

9.12.4 Validation of fairness control messages

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

This following paragraph probably belongs in clause 6 and should be generalized for all control messages.

If the source of the rcvd_fair_rate is the same station that received it then the rcvd_fair_rate will be disregarded. When comparing the rcvd_fair_rate source address, the ring identifier of the fairness control message must match the receiver's ring identifier in order to qualify as a valid compare. The exception is if the receive station is in the wrap state, in which case the fairness control message's ring identifier is ignored.

9.12.5 Packet format

The fairness control message format is as shown in Figure 9.6. The fairness protocol determines the number of hops to the station that originated the fairness control message by ~~(256 -~~ subtracting the TTL in the received ~~message)~~message from 256. Therefore all fairness control messages shall be sourced with a TTL of 255. And all single choke fairness control messages shall reset the TTL to 255 upon changing the value of the SA to the local SA.

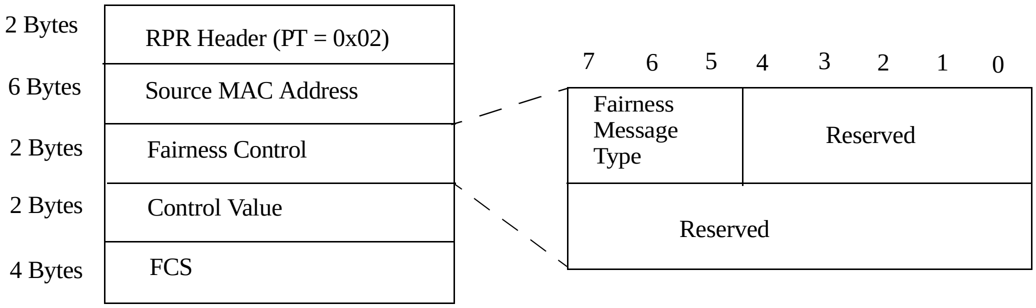


Figure 9.5—Fairness Packet Format

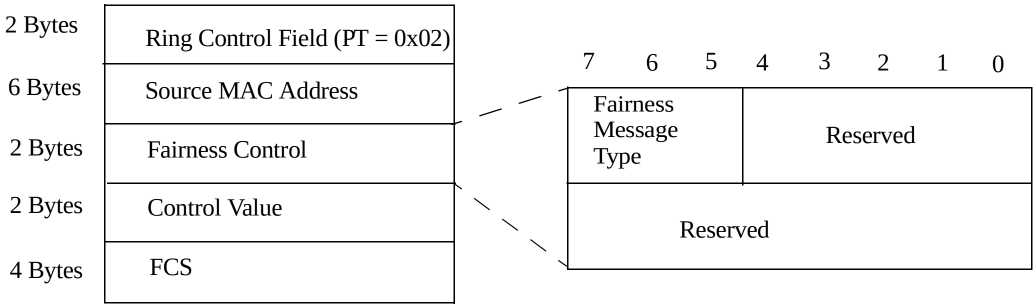


Figure 9.6—fairness control message Format

9.12.5.1 Fairness Control Message Type (3 bits)

This field specifies the type of fairness control message. Table 9.7 shows the values of the Fairness Control Message Type. ~~Type A~~ Single Choke fairness control messages are used to implement RPR-FA and provide information about the most congested span on the ringlet. ~~Type B~~ Multi Choke fairness control messages are needed to support a multi-choke implementation.

9.12.5.2 Reserved field (~~12~~ 13 bits)

These bits are ignored on receipt and set to zero on transmit for both ~~Type A~~ Single Choke and ~~Type B~~ Multi Choke fairness control messages.

9.12.5.3 Control value (16 bits)

This field carries the fair rate encoded as a 16-bit quantity. A fair rate of all ones indicates a value of ~~NULL~~ FULL_RATE which corresponds to advertising a fair rate equal to the line rate. The fair rate, when not ~~NULL~~FULL_RATE, indicates the number of bytes that ~~the~~ a station ~~was able to~~ may add to the ringlet during the ~~last decay interval divided~~ next aging interval. The reported fair rate is a rate that has been normalized by the normalization coefficient (NORMCOEF), to normalize the number of aging intervals incorporated into the rate (AGECOEF), the local station ~~weight. To ensure that this value does not overflow~~

Table 9.7—Fairness Control Message Type values

Value (binary)	Type of fairness control message	How is it used
000	~~Type A~~Single Choke	Generated by the FCU every ~~advertisement interval~~advertisement_interval. SA is the MAC address of the most congested ~~node~~ station in the fairness domain
001	~~Type B~~Multi Choke	Generated by the FCU every 10 ~~advertisement intervals~~advertisement_intervals. The SA is the SA of the MAC, and the messages are broadcast. The received fair rate is passed to the client.
010 to 111	Reserved	For future use.

~~16 bits within a decay_interval~~weight (WEIGHT), ~~a normalization factor is used depending on~~ and the ~~link speed as shown inTable 9.2~~rate coefficient (RATECOEF).

Table 9.8—Normailization Factors

Link Speed	Normalization factor
Up to and including OC-48	1
OC-192	4
OC-768	16

9.13 Fairness algorithm formulas

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

A large amount of the following section probably belongs in clause I. Some of the following will probably remain as equations inlined in the text to augment descriptions that need a more precise definition.

This state machines in this clause use the following formulas:

9.13.1 Initialization

At the beginning of the state machine, initialize:

- allowed_rate
- full_threshold
- high_threshold
- low_threshold
- unreserved_rate

```
allowed_rate = MAX_ALLOWED_RATE;

unreserved_rate = LINK_RATE - reserved_rate;

if (dual_queue_MAC)
    full_threshold = STQ_SIZE - MTU_SIZE; // can subtract more than MTU_SIZE

if (dual_queue_MAC && aggressive)
    high_threshold = STQ_SIZE / 4;
else
    high_threshold = .95 * unreserved_rate;

if (dual_queue_MAC && aggressive)
    low_threshold = STQ_SIZE / 8;
else
    low_threshold = .8 * unreserved_rate;
```

9.13.2 Per byte

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

This section (per byte operations) should be moved to Clause 6.

For each byte transmitted, update:

- add_rate
- add_rate_congested
- fw_rate
- fw_rate_congested
- nr_xmit_rate

```
if (FE byte added by local station)
    add_rate = add_rate + 1;
if (FE byte added by local station && sent beyond congestion_point)
    add_rate_congested = add_rate_congested + 1;
if (FE byte forwarded by local station)
    fw_rate = fw_rate + 1;
if (FE byte forwarded by local station && sent beyond congestion_point)
    fw_rate_congested = fw_rate_congested + 1;
if (non-A0 byte transmitted by local station)
    nr_xmit_rate = nr_xmit_rate + 1;
```

9.13.3 Per aging_interval

For each aging_interval, based upon the last received FCM, update:

- active_stations
- add_rate
- add_rate_ok
- add_rate_congested
- add_rate_congested_ok
- fw_rate
- fw_rate_congested
- nr_xmit_rate
- congested
- allowed_rate
- allowed_rate_congested
- congestion_point
- TTL_to_congestion
- local_fair_rate
- norm_local_fair_rate
- lp_add_rate
- lp_add_rate_congested
- lp_fw_rate
- lp_fw_rate_congested
- lp_nr_xmit_rate
- norm_lp_fw_rate_congested

9.13.3.1 active_stations

```
for (active_stations = 0; station = 0; station < num_stations; station++)
    if (sent_FE_during_aging_interval(station))
        active_stations = active_stations + 1;
```

9.13.3.2 lp_add_rate, lp_add_rate_congested, lp_fw_rate, lp_fw_rate_congested, lp_nr_xmit_rate, norm_lp_fw_rate_congested

The following low pass filterings must be done before the first pass agings specified in 9.13.3.3 .

```
lp_add_rate = ((LPCOEF-1) * lp_add_rate + add_rate) / LPCOEF;
lp_add_rate_congested = ((LPCOEF-1) * lp_add_rate_congested +
    add_rate_congested) / LPCOEF;
lp_fw_rate = ((LPCOEF-1) * lp_fw_rate + fw_rate) / LPCOEF;
lp_fw_rate_congested = ((LPCOEF-1) * lp_fw_rate_congested +
    fw_rate_congested) / LPCOEF;
lp_nr_xmit_rate = ((LPCOEF-1) * lp_nr_xmit_rate + nr_xmit_rate) / LPCOEF;
norm_lp_fw_rate_congested = lp_fw_rate_congested / NORMCOEF;

// alternatively, each can be simplified as shown for lp_add_rate
// lp_add_rate = lp_add_rate -
//     lp_add_rate>>(log2(LPCOEF)) + add_rate>>(log2(LPCOEF));
```

9.13.3.3 add_rate, add_rate_congested, fw_rate, fw_rate_congested, nr_xmit_rate

The following first pass agings must be done after the low pass filterings specified in 9.13.3.2 .

```
add_rate = (add_rate * (AGECOEF - 1)) / AGECOEF;
add_rate_congested = (add_rate_congested * (AGECOEF - 1)) / AGECOEF;
fw_rate = (fw_rate * (AGECOEF - 1)) / AGECOEF;
fw_rate_congested = (fw_rate_congested * (AGECOEF-1)) / AGECOEF;
nr_xmit_rate = (nr_xmit_rate * (AGECOEF - 1)) / AGECOEF;
```

9.13.3.4 congested

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

This does not currently take into account any hyteresis. This needs to be added. The hyteresis should allow delta to be set to a range of values including 0 (no hyteresis).

```
if (dual_queue_MAC)
{
    if (lp_nr_xmit_rate > unreserved_rate)
    || (STQ_depth > low_threshold))
        congested = TRUE;
    else if (aggressive) // clearing of congested for conservative mode
        congested = FALSE; // is handled in calculation for local_fair_rate
}
else // mono_queue_MAC
{
    if (lp_nr_xmit_rate > unreserved_rate)
    || (lp_nr_xmit_rate > low_threshold)
    || (access delay timer for add class B expired)
    || (access delay timer for add class C expired))
        congested = TRUE;
    else if (aggressive) // clearing of congested for conservative mode
        congested = FALSE; // is handled in calculation for local_fair_rate
}
```

9.13.3.5 allowed_rate

```
if (aggressive)
    allowed_rate = MAX_ALLOWED_RATE;
else // conservative
{
    if (congested) // local station is congested
            allowed_rate = local_fair_rate;
    else // local station is not congested
        allowed_rate = allowed_rate+(MAX_ALLOWED_RATE-allowed_rate)/RAMPCOEF;
}
```

9.13.3.6 allowed_rate_congested, congestion_point, TTL_to_congestion

```
if (rcvd_fair_rate != FULL_RATE)
{
    allowed_rate_congested = rcvd_fair_rate * NORMCOEF;
    congestion_point = rcvd_SA;
    TTL_to_congestion = 256 - rcvd_TTL;
}
else // no congestion downstream, ramp up
{
    allowed_rate_congested = allowed_rate_congested +
        (MAX_ALLOWED_RATE - allowed_rate_congested) / RAMPCOEF;
    congestion_point = local_SA;
    TTL_to_congestion = TTL_to_congestion;
    // TTL_to_congestion should be left at the last congestion point so that
    // the allowed_rate is not all of a sudden presented to the former
    // congestion point. One could add the following statement:
    // if (allowed_rate_congested == allowed_rate)
    //      TTL_to_congestion = 255;
    // But it is unnecessary since it has no effect upon the rate at which
    // the station may transmit.
}
```

9.13.3.7 local_fair_rate, norm_local_fair_rate

```
if (aggressive)
{
    if (congested)
        local_fair_rate = lp_add_rate;
    else // not congested
        local_fair_rate = unreserved_rate;
}
else // conservative
{
    if (congested)
        if (just_entered_congested) // first time when congested
            local_fair_rate = (unreserved_rate / active_stations) * WEIGHT;
            // optionally, instead of active_stations, which uses a weight of 1
            // for each of the other stations, one could use sum(ActiveWeights)
        else // was congested last time
        {
            if ((add_rate + fw_rate < low_threshold)
            && (RTT worth of aging_intervals passed since last update))
            {
                local_fair_rate = min(unreserved_rate,
                    local_fair_rate +
                    (unreserved_rate - local_fair_rate) / RAMPCOEF;
                if (local_fair_rate >= unreserved_rate)
                    congested = FALSE;
            }
            else if ((add_rate + fw_rate > high_threshold)
            && (RTT worth of aging_intervals passed since last update))
                local_fair_rate = local_fair_rate - local_fair_rate / RAMPCOEF;
            else
                local_fair_rate = local_fair_rate; // no change
        }
    else // not congested
        local_fair_rate = unreserved_rate;
}

norm_local_fair_rate = local_fair_rate / NORMCOEF;
```

9.13.3.8 add_rate_ok, add_rate_congested_ok

```
add_rate_ok =
        (add_rate < allowed_rate) \\ current allowance is not exceeded
    && (lp_nr_xmit_rate < unreserved_rate) \\ space for reserved traffic
    && ((STQ_depth = 0) \\ upstream stations are not in need
        || ((fw_rate > add_rate) \\ upstream stations are not starved
            && (STQ_depth < high_threshold))); \\ node is not fully congested
```

NOTE—An optional optimization is to change (fw_rate > add_rate) to (fw_rate > add_rate/WEIGHT). This is not needed under almost all circumstances. The one exception is if the sum of weights of upstream stations is less than the weight of the local station. If this optimization is chosen, the implementation may require that WEIGHT values be factors of 2.

```
   add_rate_congested_ok = add_rate_ok
                                   && (add_rate_congestion < allowed_rate_congestion);
```

9.13.3.9 sendC

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

This section (sendC) should be moved to Clause 6.

```
   if (!add_rate_ok)
      sendC = 0;
   else if (!add_rate_congested_ok)
      sendC = TTL_to_congestion;
   else                             // see comment in 9.13.3.6 on TTL_to_congestion
      sendC = TTL_to_congestion; // calculation with no downstream congestion
```

9.13.4 Per advertisement_interval

For each advertisement_interval, update:

— advertised_fair_rate

```
   if ((rcvd_fair_rate < norm_local_fair_rate) // downstream is more congested,
   && (rcvd_fair_rate < norm_lp_fw_rate_congested)) // upstream is contributing
      advertised_fair_rate = rcvd_fair_rate;          // to downstream congestion
   else if (congested) // local station is congested (and more than downstream)
      advertised_fair_rate = norm_local_fair_rate;
   else // no downstream or local congestion
      advertised_fair_rate = FULL_RATE;
```