



**IEEE Standard for Information technology—
Telecommunications and information exchange between systems—
Local and metropolitan area networks—
Specific requirements**

Part 17: Resilient packet ring (RPR) access method and physical layer specifications

Amendment 1: Spatially aware sublayer

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
23 July 2007

IEEE Std 802.17b™-2007
(Amendment to
IEEE Std 802.17™-2004)

**IEEE Standard for Information technology—
Telecommunications and information exchange between systems—
Local and metropolitan area networks—
Specific requirements**

Part 17: Resilient packet ring (RPR) access method and physical layer specifications

Amendment 1: Spatially aware sublayer

Sponsor

**LAN/MAN Standards Committee
of the
IEEE Computer Society**

Approved 8 March 2007

IEEE SA-Standards Board

Abstract: This amendment enables a service provider to use the architecture and protocols of IEEE Std 802.17™-2004 to offer improvements in bandwidth utilization, by means of spatial reuse, for applications of RPR that involve bridging clients.

Keywords: bridging, fairness, layer management, medium access control, metropolitan area network (MAN), physical layer, protection switching, resilient packet ring (RPR), ring topology, spatial reuse, topology detection

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2007 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 23 July 2007. Printed in the United States of America.

IEEE and 802 are registered trademarks in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

Print: ISBN 0-7381-5521-7 SH95626
PDF: ISBN 0-7381-5522-5 SS95626

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied “**AS IS**.”

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

This introduction is not part of IEEE Std 802.17b-2007, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 17: Resilient packet ring (RPR) access method and physical layer specifications—Amendment 1: Spatially aware sublayer.

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/updates/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/interp/index.html>.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. A patent holder or patent applicant has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates and nondiscriminatory, reasonable terms and conditions to applicants desiring to obtain such licenses. The IEEE makes no representation as to the reasonableness of rates, terms, and conditions of the license agreements offered by patent holders or patent applicants. Further information may be obtained from the IEEE Standards Department.

Participants

The following is a list of participants in the 802.17 Working Group.

Michael L. Takefman, *Chair*

John Lemon, *Vice Chair*

Marc Holness, *Editor*

Thomas Alexander
Leon Bruckman
Robert Castellano
Nitin Gogate
John F. Hawkins

David V. James
Peter G. Jones
Michael Kelsen
Glenn W. Parsons
Harry Peng

Tim Plunkett
Raj Sharma
Robert Sultan
Gary Turner
George Young

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Eng Ahmed Abdelhalim
Toru Aihara
Thomas Alexander
Butch Anton
Vinay K. Bannai
John B. Barnett
Hugh Barrass
Gennaro Boggia
Leon Bruckman
Matthew K. Burnburg
William A. Byrd
Edward J. Carley Jr.
James T. Carlo
Juan C. Carreon
Danila Chernetsov
Keith Chow
Tommy P. Cooper
Thomas J. Dineen
Sean Dougherty
Sourav K. Dutta
Donald W. Fedyk
John W. Fendrich
Brian G. Floyd

Devon L. Gayle
Sergiu R. Goma
Randall C. Groves
Robert M. Grow
John F. Hawkins
Dennis Horwitz
Atsushi Ito
Raj Jain
David V. James
Anthony A. Jeffree
Peter G. Jones
Shinkyō Kaku
Junghong Kao
Piotr Karocki
Stuart J. Kerry
Yongbum Kim
Juan L. Lazaro
John Lemon
William Lumpkins
G. L. Luri
Jonathon C. McLendon
Francisco J. Melendez

George J. Miao
Gary L. Michel
Scott Misha
William J. Mitchell
Michael S. Newman
Satoshi Obara
Glenn W. Parsons
Subburajan
Ponnuswamy
Vikram Punj
Robert A. Robinson
Bartien Sayogo
Stephen C. Schwarm
Raj Sharma
Manikantan Srinivasan
Thomas E. Starai
Robert Sultan
Michael L. Takefman
Patricia A. Thaler
Mark-Rene Uchida
Takahito Yoshizawa
George Young
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 8 March 2007, it had the following membership:

Steve M. Mills, *Chair*
Robert M. Grow, *Vice Chair*
Don Wright, *Past Chair*
Judith Gorman, *Secretary*

Richard DeBlasio
Alex Gelman
William R. Goldbach
Arnold M. Greenspan
Joanna N. Guenin
Julian Forster*
Kenneth S. Hanus
William B. Hopf

Richard H. Hulett
Hermann Koch
Joseph L. Koepfinger*
John Kulick
David J. Law
Glenn Parsons
Ronald C. Petersen
Tom A. Prevost

Narayanan Ramachandran
Greg Ratta
Robby Robson
Anne-Marie Sahazizian
Virginia C. Sulzberger
Malcolm V. Thaden
Richard L. Townsend
Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Alan H. Cookson, *NIST Representative*

Don Messina
IEEE Standards Program Manager, Document Development

Michael D. Kipness
IEEE Standards Program Manager, Technical Program Development

Contents

1.	Overview	2
2.	Document structure	2
3.	Terms, definitions, and notation	3
3.2	Terms and definitions	3
4.	Abbreviations and acronyms	5
5.	Architecture overview	7
5.10	Spatial reuse	7
5.16	Frame ordering	7
5.16.1	Use of italics	7
5.17	Operations, administration, and maintenance (OAM)	7
5.18	Spatially aware sublayer	7
6.	Medium access control (MAC) service and reference model	9
6.2	Terminology and variables	9
6.3	Overview of MAC services	9
6.3.1	Service types	9
6.4	MAC services to the client layer	11
6.4.2	MA_DATA.indication	14
6.4.3	MA_CONTROL.request	14
6.4.4	MA_CONTROL.indication	15
6.6	MAC reference model	15
6.6.1	MAC control sublayer	15
6.6.2	MAC datapath sublayer	17
6.6.3	Flow of data within the MAC	17
7.	Medium access control datapath	19
7.1	Datapath overview	19
7.2	Terminology and variables	19
7.2.1	Common state machine definitions	19
7.2.3	Common state machine routines	20
7.2.4	Variables and literals defined in other clauses	20
7.2.5	Flow count variables	20
7.6	Receive operation	20
7.7	Transmit operation	26
7.7.1	Ringlet selection	26
7.7.2	Determination of cleave point	29
7.7.3	Setting of <i>ttl</i> and <i>ttlBase</i>	30
7.7.6	ControlAddCount state machine	33
7.8	Protocol Implementation Conformance Statement (PICS) proforma for Clause 7	35
7.8.3	PICS tables for Clause 7	35

9.	Frame formats	37
	9.3 Control frame format	37
	9.3.2 Control frame fields	37
11.	Topology discovery and protection	39
	11.2 Terminology and variables	39
	11.2.5 <i>myTopoInfo</i> fields	39
	11.2.6 <i>topoEntry[rid][hops]</i> fields	39
	11.2.7 Common state machine routines	39
	11.4 Defined ATT encodings.....	39
	11.4.3 Station settings ATT	39
	11.5 Topology database	40
	11.5.1 Topology database structure	40
	11.6 State machines	40
	11.6.6 TopologyValidation state machine	40
12.	Operations, administration, and maintenance (OAM).....	41
	12.1 Overview	41
	12.1.2 OAM functions supported by RPR	41
	12.1.7 SAS notify operations	41
	12.2 Terminology and variables	41
	12.2.3 Common routines	41
	12.2.4 Literals and routines defined in other clauses	42
	12.3 OAM frame formats	42
	12.3.4 SAS notify frame	42
	12.4 OAM service primitives	44
	12.4.6 Organization specific MA_CONTROL.indication	44
	12.4.7 SAS notify MA_CONTROL.request	44
	12.5 OAM state machines	45
	12.5.1 OamFrameTransmit state machine	45
	12.5.2 OamFrameReceive state machine	45
	12.7 Protocol Implementation Conformance Statement (PICS) proforma for Clause 12	46
	12.7.3 PICS tables for Clause 12	46
14.	Spatially aware sublayer	47
	14.1 Spatially aware sublayer overview	47
	14.2 Terminology and variables	48
	14.2.1 Common definitions	48
	14.2.2 Common state machine variables	48
	14.2.3 Common state machine routines	49
	14.2.4 Variables and literals defined in other clauses	49
	14.2.5 SAS database fields	50
	14.2.6 SAS statistics counters	52
	14.3 SAS association database.....	52
	14.3.1 Logical representation of SAS tables	52
	14.3.2 Static unicast entries	54
	14.3.3 Static multicast entries	54
	14.3.4 Dynamic entries	54
	14.3.5 Mapping of VIDs to FIDs	55

14.4 SDB operations	55
14.4.1 SAS learning process	55
14.4.2 Querying the SDB	56
14.4.3 SDB entry removal	56
14.5 SAS operations	57
14.5.1 Principles of unicast operation	57
14.5.2 Principles of multicast operation	58
14.5.3 Transmit operation	60
14.5.4 SAS receive operation	64
14.6 Protocol Implementation Conformance Statement (PICS) proforma for Clause 14	67
14.6.1 Introduction	67
14.6.2 Identification	67
14.6.3 PICS tables for Clause 14	68
Annex D (normative) SNMP MIB definitions	71
D.3 Security considerations	71
D.4 MIB Structure	72
D.4.1 Structure of the MIB	72
D.6 Definitions for the RPR MIB	72

Information technology—
Telecommunications and information exchange between
systems—
Local and metropolitan area networks—
Specific requirements

Part 17: Resilient packet ring (RPR) access method and physical layer specifications

Amendment 1: Spatially aware sublayer

NOTE—The editing instructions contained in this amendment define how to merge the material contained therein into the existing base standard and its amendments to form the comprehensive standard.

The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~striketrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editorial notes will not be carried over into future editions because the changes will be incorporated into the base standard.

1. Overview

1.4 Document structure

Insert the following item in the list of clauses, after item m). Renumber as necessary.

- n) Clause 14: Spatially aware sublayer

3. Terms, definitions, and notation

3.2 Terms and definitions

Insert the following definitions in alphabetical order into 3.2, renumbering as necessary.

3.2.1 directed transmission: Transmission of a frame to a single station on the medium.

3.2.2 permissive: Frame transmission that permits (within a conversation) frame reorder and duplication while recovering from failures on the ring, and frame reorder when changing between undirected and directed transmission.

3.2.3 relaxed: Frame transmission that permits frame reorder and duplication within a conversation when recovering from failures on the ring.

3.2.4 ring local address: The station address of one station on the ring.

3.2.5 remote address: Any unicast address that is not a ring local address.

3.2.6 scoped transmission: A method of transmitting a multicast frame with a TTL value that limits its reach to the furthest station that requires the frame.

3.2.7 strict: Frame transmission that prevents frame reorder and duplication within a conversation.

3.2.8 undirected transmission: Transmission of a frame on to the medium using flooding.

4. Abbreviations and acronyms

Insert the following abbreviations in alphabetical order into Clause 4:

FDB	filtering database
FID	FDB identifier
MSTI	Multiple Spanning Tree Instance
SAS	spatially aware sublayer
SDB	SAS database
VID	VLAN identifier

5. Architecture overview

5.10 Spatial reuse

Insert the following paragraph at the end of 5.10.

Spatial reuse for non-local frame transmission is supported by the optional spatially aware sublayer (see 5.18).

5.16 Frame ordering

5.16.1 Frame transmission types

Change the text found in 5.16.1 as follows:

~~Two~~Three types of frame transmission are supported: permissive, relaxed, and strict. The adherence to the strict requirements (see 6.3 and 7.6.1) by the MAC is not only applicable to MACs servicing IEEE Std 802.1D™-2004 and IEEE Std 802.1Q™-2003 compliant clients (i.e., bridges), but also MACs servicing other clients (e.g., host, router, etc.). The requirements of strict transmissions are as follows:

- a) Duplication of user data frames is not permitted.
- b) Reordering of frames that have identical values of the following three parameters is not permitted:
 - 1) MAC address of the destination.
 - 2) MAC address of the source.
 - 3) Service class of the frame.

The complexity of supporting strict transmission is particularly burdensome during station or link failures. With relaxed and permissive mode, some reorder and/or duplication is possible, but frame losses can be decreased. With permissive mode, some reorder is also possible when the spatially aware sublayer (SAS) transitions between undirected and directed transmissions for a conversation.

The processing of strict ~~and~~, relaxed, and permissive transmissions is similar, but strict frames are discarded more frequently. The intent is to discard suspect frames, rather than risk the possibility of duplicating or misordering frames with distinct old-topology and new-topology heritages. Consistency checks and discard strategies handle passthrough (see Figure 5.25-b) and severed span (see Figure 5.25-c through Figure 5.25-f) failure scenarios.

5.17 Operations, administration, and maintenance (OAM)

Insert 5.18 and accompanying text after 5.17.3.

5.18 Spatially aware sublayer

SAS is a sublayer of the MAC datapath sublayer. SAS functionality is optional and provides spatial reuse for frame transmissions other than local unicast (e.g., through bridges on the ring), by using directed transmissions or multicast scoping. If SAS functionality is not implemented, *enableSas* (see 6.2.4) is always FALSE, resulting in no spatial awareness for frame transmissions other than local unicast.

6. Medium access control (MAC) service and reference model

6.2 Terminology and variables

Delete 6.2.2.

Insert the following terms and variables in alphabetical order into 6.2. Renumber as necessary.

6.2.1 *enableSas*: A boolean variable determining whether SAS processing is enabled. A value of TRUE enables processing of the `request_sas` parameter of the `MA_DATA.request` primitive, and learning of address associations in SAS receive state tables. A value of FALSE causes all data frames to bypass SAS. The method of configuring *enableSas* is not standardized.

6.2.2 *optionSasMcastScope*: A boolean variable that controls the optional SAS multicast scoping functionality. A value of TRUE indicates that SAS multicast scoping is active. A value of FALSE indicates that SAS multicast scoping is not active. A value of TRUE can be set only if *enableSas* is set to TRUE. The default value is FALSE. The method of configuring *optionSasMcastScope* is not standardized.

6.2.3 *remapRxAddr*: A boolean variable that controls the mapping of extended addresses in the MAC receive path. A value of TRUE means the MAC receive path sets `source_address` to `source_address_extended`, and `destination_address` to `destination_address_extended`. A value of FALSE indicates that the MAC receive path does not modify the address parameters. The default value is TRUE. The method of configuring *remapRxAddr* is not standardized.

6.2.4 *stdCleave*: A boolean variable indicating whether a SAS-capable station should use bidirectional flooding and the standard cleave point calculation. A value of TRUE causes all flooded data frames to use the standard cleave point calculation per 7.7.2.1. A value of FALSE indicates that neither the standard cleave point calculation nor bidirectional flooding need to be applied by the MAC. The method of configuring *stdCleave* is not standardized.

6.2.5 *transmissionControl*: Determines the transmission service provided by the MAC.

STRICT—The MAC uses the strict transmission service for all frames unless the client sets the `strict_order` parameter to FALSE; in which case, the frame uses the relaxed transmission service.

RELAXED—The MAC uses the relaxed transmission service for all frames unless the client sets the `strict_order` parameter to TRUE; in which case, the frame uses the strict transmission service.

PERMISSIVE—The MAC uses the permissive transmission service for all frames unless the client sets the `strict_order` parameter to TRUE; in which case, the frame uses the strict transmission service. PERMISSIVE can be set only if *enableSas* is set to TRUE.

The default is STRICT. The method of configuring *transmissionControl* is not standardized.

6.3 Overview of MAC services

6.3.1 Service types

Change the text of 6.3.1 as follows:

The RPR MAC provides ~~two~~ three types of frame transmission service, described below. (See also 7.6.1, Table 7.20, and Table 7.29.)

a) Strict: Adheres to the 802.1 frame reorder, duplication, and loss requirements, as follows:

1) There is no guarantee that all service data units (SDUs) are delivered.

- 2) Reordering of frames with a given user priority for a given combination of destination address and source address is not permitted.
- 3) Duplication of user data frames is not permitted.

In general, strict transmission requires special processing only during station or link failure events.

- b) Relaxed: Adheres to the 802.1 frame reorder, duplication, and loss requirements, except while recovering from failures on the ring. During ring protection events, a minimal amount of reorder and/or duplication can be encountered; however, the chance of delivery is increased.

Scenarios where a negligible amount of frame reorder or duplication can occur include the following:

- 1) After ring (link or station) restoration events.
 - 2) Topology and status database of stations on the ring are not synchronized.
 - 3) Station failure resulting in passthrough behavior. That is, frames are sent through the transit path without *ttl* decrement or any other frame processing/stripping rules being applied.
 - 4) Compound ring (link or station) failures resulting in segmented open rings.
 - 5) Rapid cascading ring failures.
- c) Permissive: Permissive has the same attributes as the relaxed transmission service, except that some reordering can occur during transitions between undirected and directed transmission for a conversation.

NOTE—Relaxed and permissive modes ~~is~~ are more efficient (i.e., fewer frames are discarded) and ~~should therefore be~~ are used when strict mode is not required.¹

The MAC sublayer presents a service interface for the exchange of MAC service data units between MAC client entities. The MAC service interface supports service classes denoted classA, classB, and classC (see 7.3 and 9.6.4).

For all service classes, the MAC service interface provides per-ringlet indications to the MAC client that indicate whether traffic can or cannot currently be accepted. For service class classC, the MAC service interface also provides the number of hops to the nearest congested station. Each service class is rate controlled to prevent the client from transmitting more traffic than was allocated by station management or allowed by fairness, as applicable. When a client does not obey the indication of the MAC with regard to flow control on a given service class, the MAC is free to ensure correct ring operation by either dropping the offending client requests or by head-of-line blocking until the request can be properly handled.

The allocation, advertisement, and shaping of allocated bandwidth is done on a per-class, per-ringlet basis.

The MAC does not provision bandwidth. This is a higher level management function beyond the scope of this standard.

The information that flows between the MAC and the client layer is specified formally in the primitives (see 6.4).

¹Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

6.4 MAC services to the client layer

6.4.1.2 Semantics of the service primitive

Change the text of 6.4.1.2 as follows:

The semantics of the primitives are as follows:

```
MA_DATA.request
(
    destination_address,
    source_address,           // optional
    mac_service_data_unit,
    frame_check_sequence,    // optional
    service_class,
    ringlet_id,              // optional
    mac_protection,          // optional
    mark_fe,                 // optional
    strict_order,            // optional
    destination_address_extended, // optional
    source_address_extended,  // optional
    flooding_form,           // optional
    request_sas             // optional
)
```

The parameters of the MA_DATA.request are described as follows:

destination_address

Specifies either an individual or group MAC address, different from the local MAC address, to be used to create the *da* (destination MAC address) field or *daExtended* (destination MAC address, extended) field of the transmitted frame (see 6.4.1.5 and 9.2.2.3).

source_address

If present, and different from the station's MAC address, specifies an individual MAC address to be used to create the *saExtended* (source MAC address, extended) field of the transmitted frame (see 9.2.2.9), and to determine basic or extended data frame format (see 6.4.1.5 and 7.7.1). The *source_address* parameter is ignored if the *source_address_extended* parameter is provided.

mac_service_data_unit

Provides the payload to be delivered. Specifically, the *mac_service_data_unit* parameter specifies the MAC service data unit to be transmitted by the MAC sublayer entity in the *serviceDataUnit* field of the transmitted frame (see 9.2.2.11). Sufficient information is associated with *mac_service_data_unit* for the MAC sublayer entity to determine the length of the data unit.

frame_check_sequence

Provides the value of the *fcs* field of the transmitted frame (see 9.2.2.12). If the *frame_check_sequence* parameter is omitted, the MAC calculates the *fcs* value (see E.2).

service_class

Indicates the class of service requested by the MAC client, as described in Table 6.1, which is used by the MAC entity to select the value of the *sc* field (see 9.6.4), and to indicate the requested MAC treatment of the transmitted frame (see 7.7.4).

ringlet_id

Indicates the ringlet choice of the client, as described in Table 6.2 (see also 7.7.1), which is used by the MAC entity to select the value of the *ri* (ringlet identifier) field (see 9.6.1) of the transmitted frame (before any protection-based change).

Table 6.1—service_class values

service_class value	Corresponding sc value	Description
SC_CLASS_A	CLASS_A0 or CLASS_A1	classA service class
SC_CLASS_B	CLASS_B	classB service class
SC_CLASS_C	CLASS_C	classC service class

Table 6.2—ringlet_id values

ringlet_id value	Corresponding ri value	Description
RI_0	RINGLET_0	ringlet0 preferred
RI_1	RINGLET_1	ringlet1 preferred
RI_DEFAULT	—	default ringlet
(null)		

mac_protection

Indicates a choice of whether the MAC provides protection for the frame (see 7.7.1).

TRUE—The MAC provides protection for the frame.

FALSE—The MAC does not provide protection for the frame.

(null)—The default value is TRUE.

mark_fe

Indicates a request to mark and treat a frame as fairness eligible regardless of how it would have been marked or treated otherwise, guiding the MAC entity on how to set the *fe* (fairness eligible) field (see 9.6.2). The *mark_fe* parameter is valid only for requests that include a request of classB for the *service_class* and is ignored for all other requests. This is provided to allow a client to choose which of the classB marked service data units presented to the MAC are considered fairness eligible, such as when handling multiple flows of traffic, each with their own client-based rate allocations (see 7.7.4).

TRUE—The MAC forces the (classB) frame to be fairness eligible.

FALSE—The MAC determines whether the frame is fairness eligible based on *sendB*.

(null)—The default value is FALSE.

strict_order

Indicates a request to mark and treat a data frame as strict or relaxed, which is used by the MAC entity to select the value of the *so* (strict order) field (see 9.7.4). ~~The strict_order parameter is ignored, and all frames have so set to 1, if the forceStrict variable is set to TRUE (see 6.2.2).~~

TRUE—The MAC sets *so* to 1.

FALSE—The MAC sets *so* to 0.

(null)—The default value is TRUE when the *transmissionControl* variable is set to STRICT, otherwise FALSE.

destination_address_extended

Specifies either an individual or group MAC address, different from the local MAC address, to be used to create the *daExtended* field of the transmitted frame (see 9.2.2.8). If the *destination_address_extended* parameter is provided, the MAC uses the extended data frame format (see 7.7.1). Whenever the *destination_address_extended* parameter is provided, the *source_address_extended* parameter is also required to be provided (see 6.4.1.5), and the *flooding_form* parameter is also required to be provided. Use of this parameter's effect on bridging is discussed in F.1.7. This parameter is ignored when SAS processing is selected.

source_address_extended

Specifies an individual MAC address, to be used to create the *saExtended* field of the transmitted frame (see 9.2.2.9). If the *source_address_extended* parameter is provided, the MAC uses the extended data frame format (see 7.7.1). Whenever the *source_address_extended* parameter is provided, the *source_address* parameter is ignored, the *destination_address_extended* parameter is also required to be provided (see 6.4.1.5), and the *flooding_form* parameter is also required to be provided. Use of this parameter's effect on bridging is discussed in F.1.7. This parameter is ignored when SAS processing is selected.

flooding_form

Indicates a request to use a particular flooding form (see 7.7.1), as specified in Table 6.3, and to set the *fi* (flooding indication) field accordingly (see 9.7.2). A value of *FI_NONE* directs the MAC to not flood the frame. A value other than *FI_NONE* directs the MAC to flood the frame regardless of the determination that would have been made otherwise by ringlet selection, and to use the flooding form selected for the frame. If *flooding_form* is omitted, the MAC uses its default flooding behavior. The *flooding_form* parameter is required to be provided if the *destination_address_extended* and *source_address_extended* parameters are provided. Use of this parameter's effect on bridging is discussed in F.1.7. This parameter is ignored when SAS processing is selected and a SAS database (SDB) entry is found, or when *stdCleave* is enabled.

Table 6.3—flooding_form values

flooding_form value	Corresponding <i>fi</i> value	Description
FLOOD_NONE	FI_NONE	no flood
FLOOD_UNIDIR	FI_UNIDIR	unidirectional flood
FLOOD_BIDIR	FI_BIDIR	bidirectional flood
(null)	default flooding behavior selected by the MAC	

request_sas

Indicates a request to provide SAS processing. If the *enableSas* variable is set to TRUE (see 6.2.1), this parameter controls the selection of SAS processing. Otherwise, SAS processing is disabled and this parameter is ignored.

TRUE—Request SAS processing for this frame.

FALSE—Bypass SAS for this frame.

(null)—The default value is the setting of the *enableSas* variable.

6.4.1.4 Effect of receipt

Change the text of 6.4.1.4 as follows:

The receipt of the *MA_DATA.request* primitive causes the MAC entity to create a data frame or an extended data frame, fill in the fields whose values are given or determined by the parameters of this request, and pass the properly formed frame to the transmit state machines by placing it on *Q_TX_SAS* (see Figure 6.2), for transfer to the peer MAC sublayer entity or entities.

6.4.1.5 Additional comments

Change the text of 6.4.1.5 as follows:

The MAC does not reflect frames back to the client. If a client issues an MA_DATA.request primitive with a destination_address value equal to its local MAC address, the request is rejected.

The MAC does not accept requests from the client when the appropriate send indication is not present. The details of this interaction are implementation specific.

For frames not processed by SAS, the combinations of source_address and source_address_extended are summarized in Table 6.4. In the case of any ambiguity between this summary and the RingletSelection and SasTransmit state machines (see 7.7.1 and 14.5.3.1), the state machines shall take precedence.

Table 6.4—Source address value combinations

source_address_extended	source_address	Resulting frame actions
not provided	not provided	// basic data frame sa = myMacAddress;
	= myMacAddress	// basic data frame sa = myMacAddress;
	≠ myMacAddress	// extended data frame sa = myMacAddress; saExtended = source_address; da = destination_address; daExtended = destination_address;
provided	—	// extended data frame sa = myMacAddress; saExtended = source_address_extended; da = destination_address; daExtended = destination_address_extended;

6.4.2 MA_DATA.indication

6.4.2.2 Semantics of the service primitive

Replace the destination_address and source_address parameter descriptions with the following:

destination_address

Indicates the value of the da (see 9.2.2.3) or daExtended (see 9.2.2.8) field of the incoming frame, as described in 14.5.4.1.4.

source_address

Indicates the value of the sa (see 9.2.2.4) or saExtended (see 9.2.2.9) field of the incoming frame, as described in 14.5.4.1.4.

6.4.3 MA_CONTROL.request

6.4.3.2 Semantics of the service primitive

Insert the following OAM_SAS_NOTIFY opcode value information in Table 6.6, in alphabetical order.

OAM_SAS_NOTIFY	Notification of a network topology change event that affects SAS forwarding.	SAS notify specific payload and parameters	12.4.7
----------------	--	--	--------

6.4.4 MA_CONTROL.indication

6.4.4.2 Semantics of the service primitive

Insert the following OAM_SAS_NOTIFY_IND opcode value information in Table 6.7, in alphabetical order.

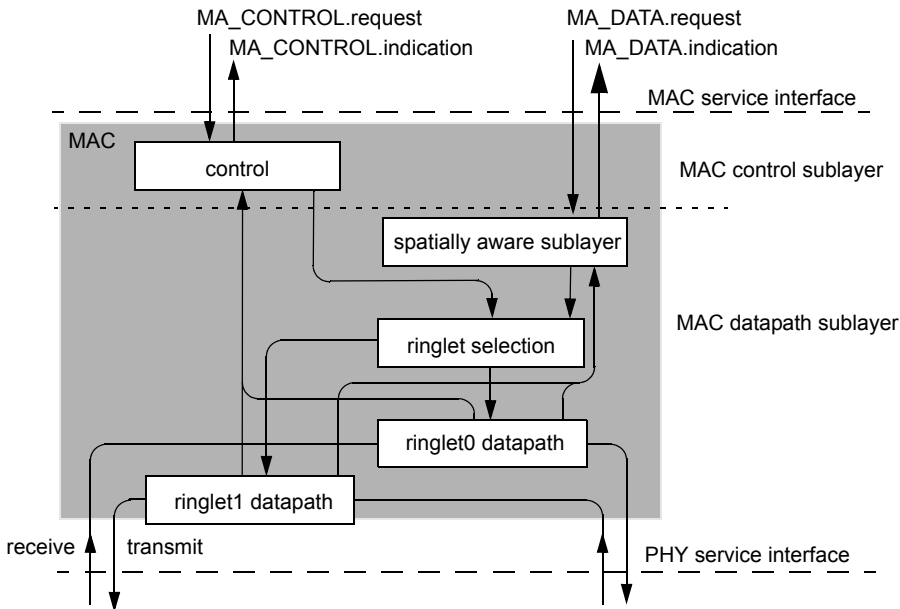
OAM_SAS_NOTIFY_IND	Receipt of a SAS notify specific OAM frame	SAS notify payload and parameters	12.5
--------------------	--	-----------------------------------	------

6.6 MAC reference model

Change the first paragraph of 6.6 as follows:

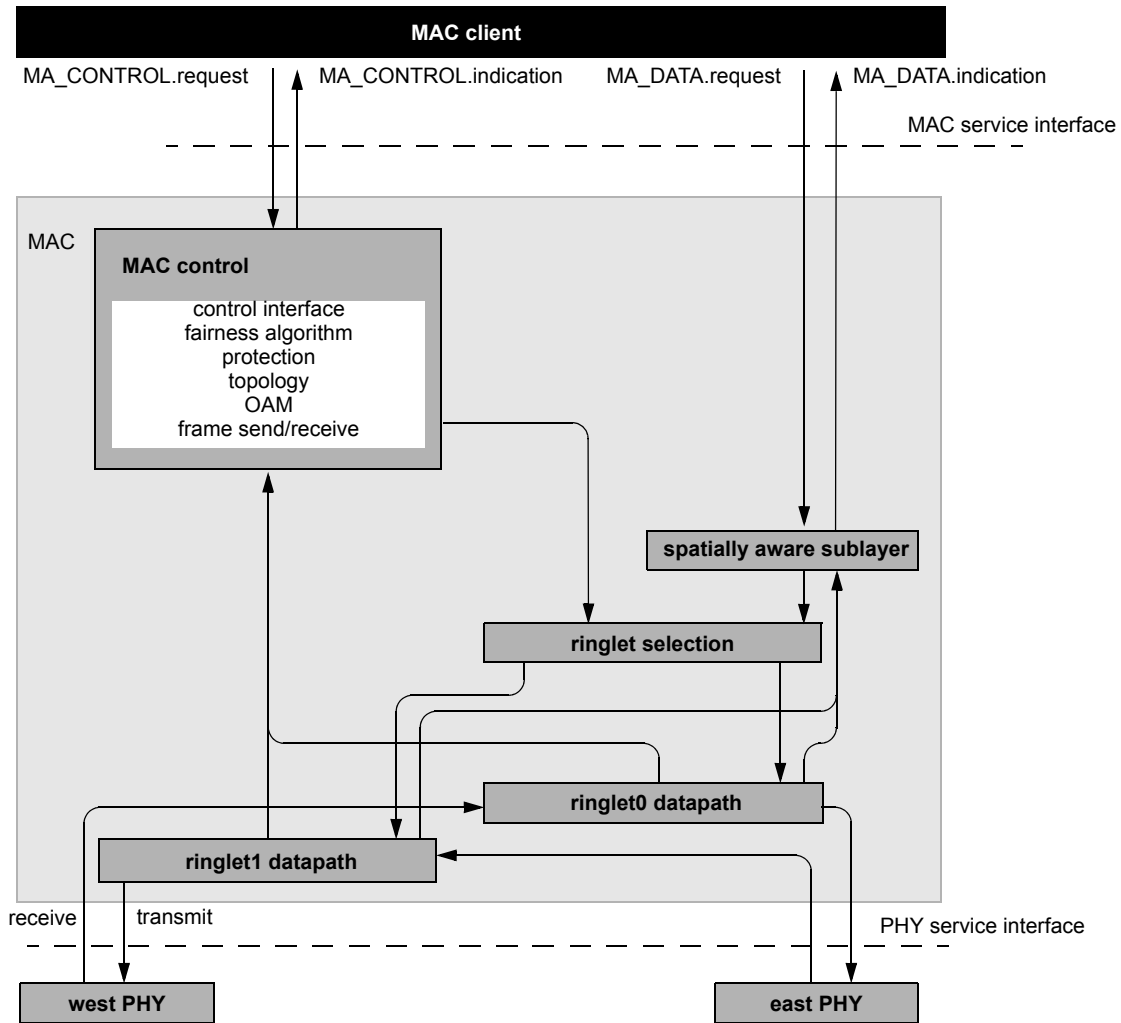
As illustrated in Figure 6.1, the MAC comprises the MAC control sublayer and the MAC datapath sublayer. The MAC datapath sublayer comprises the spatially aware sublayer, the ringlet selection entity and the datapaths for the two ringlets. These components and their interconnections are illustrated in Figure 6.4 within the context of a single station view of the MAC architecture.

Replace Figure 6.4 with the following figure:



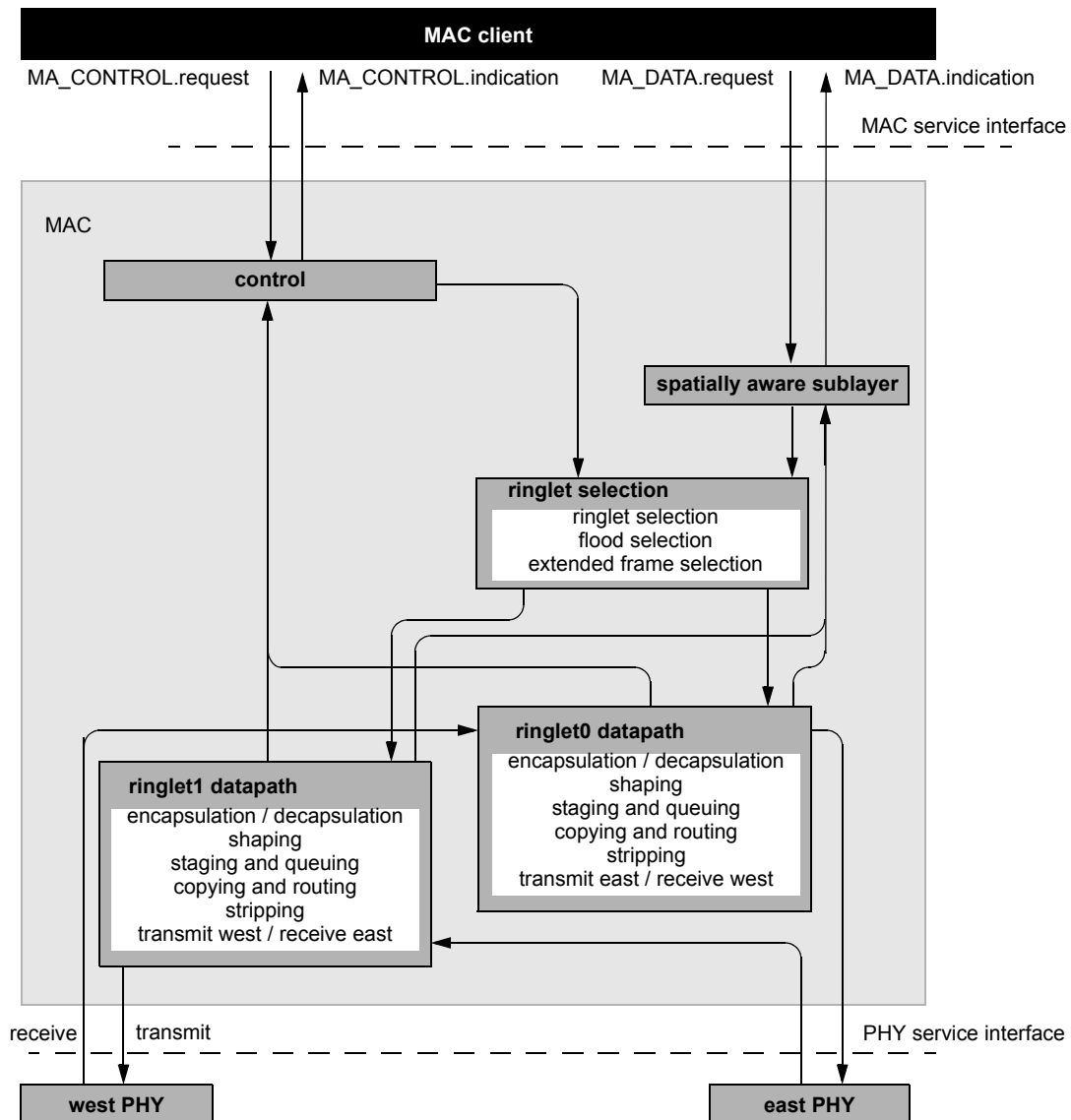
6.6.1 MAC control sublayer

Replace Figure 6.5 with the following figure:



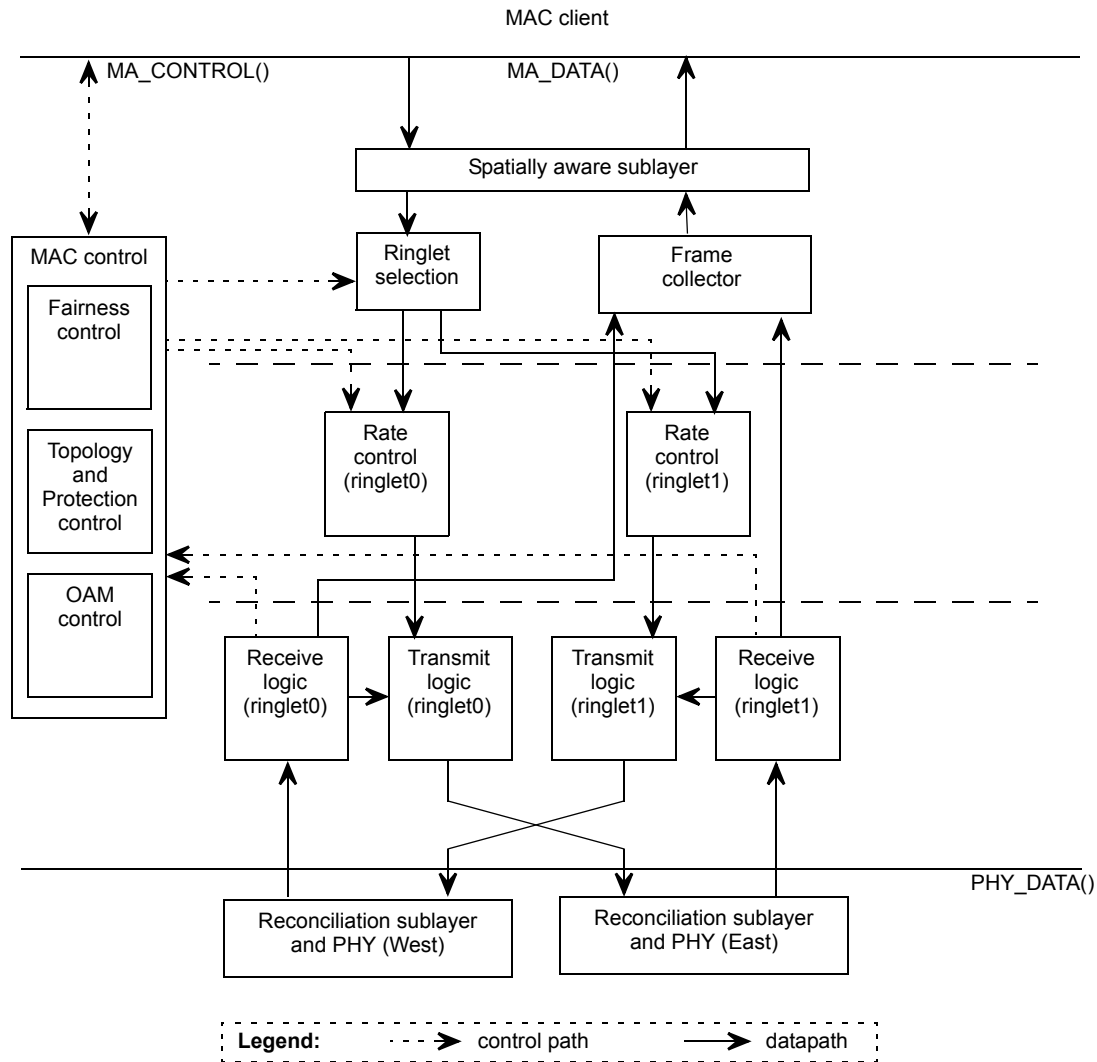
6.6.2 MAC datapath sublayer

Replace Figure 6.6 with the following figure:



6.6.3 Flow of data within the MAC

Replace Figure 6.7 with the following figure:



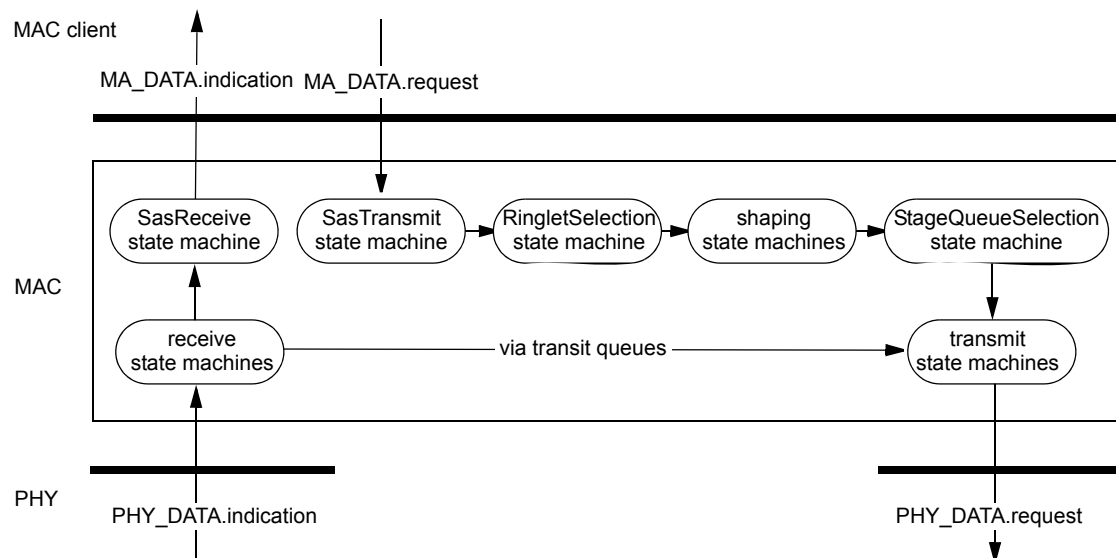
7. Medium access control datapath

7.1 Datapath overview

Change the second-to-last paragraph of 7.1 as follows:

The various pieces of the datapath sublayer are tied together through logical queues. For example, an invocation of MA_DATA.request places input into a queue leading to the ~~RingletSelection~~ SasTransmit state machine. As this and each subsequent state machine processes the input, it takes the input off of one queue and then places it onto another queue. This continues until the last queue leads to a PHY_DATA.request. Similarly, invocation of PHY_DATA.indication (by the reconciliation sublayer) provides a frame to each of the receive state machines in turn, via logical queues, and then eventually to one of the transmit state machines, via the transit queue(s), and to MA_DATA.indication. These relationships are illustrated in a simplified overview in Figure 7.2.

Replace the contents of Figure 7.2 with the following:



7.2 Terminology and variables

7.2.1 Common state machine definitions

Insert the following “queue values” into the existing “queue values” section of 7.2.1.

- Q_RX_SAS—The queue identifier associated with SAS receive processing of frames.
- Q_RX_SAS_NOTIFY—The queue identifier associated with received SAS notify frames.
- Q_TX_SAS—The queue identifier associated with SAS transmit processing of frames.

Insert the following definitions in alphabetical order into 7.2.1.

SAS_GROUP_ADDRESS

The reserved group MAC address used by SAS (see 14.5.3.1.4).
Value: 01-80-C2-00-00-04.

7.2.3 Common state machine routines

Insert the following routines in alphabetical order into 7.2.3.

Provided(parameter)

To indicate whether an optional parameter was provided in the MA_DATA.request service primitive invocation or by a sublayer of the MAC.

TRUE—The parameter was provided.

FALSE—(Otherwise.)

Reachable(address, ringlet)

To indicate whether the topology database (see 11.5) shows that the specified destination MAC address is reachable via the specified ringlet. The reachability is determined from topoEntry[ringlet][index].reachable, where index is the index value of the station entry with topoEntry[ringlet][index].macAddress equal to address.

TRUE—The destination address is reachable via the specified ringlet.

FALSE—(Otherwise.)

7.2.4 Variables and literals defined in other clauses

Remove the following variable, in the “parameters, literals, and variables defined in Clause 6” section.

forceStrict

Insert the following parameters and variables in alphabetical order, in the “parameters, literals, and variables defined in Clause 6” section.

remapRxAddrs

enableSas

optionSasMcastScope

request_sas

transmissionControl

Insert the following literal in alphabetical order, in the “literals defined in Clause 9” section.

CT_OAM_SAS_NOTIFY

7.2.5 Flow count variables

Insert the following flow count variables in alphabetical order, in 7.2.5.

fromCtrlOamSasNotifyFrames

toCtrlOamSasNotifyFrames

7.6 Receive operation

7.6.3.1 Receive entity

Replace the content of 7.6.3.1 with the following. All references in the new text refer to the new numbering.

Each station has transmit and receive entities, as illustrated in Figure 7.8, which use rules designed to support local and bridged RPR traffic.

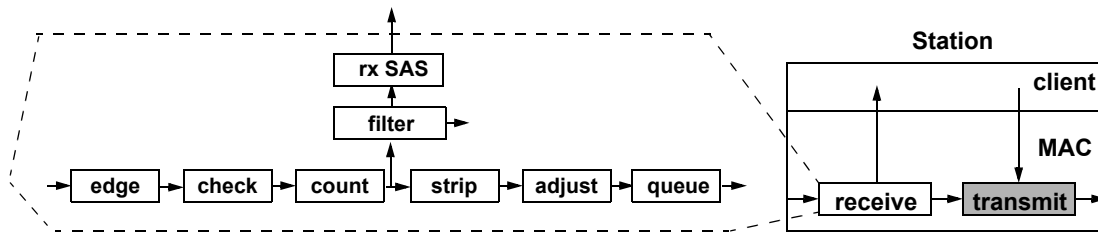


Figure 7.8—Receive subentities

The receive entity includes the following subentities, detailed in 7.6.3.3–7.6.3.10. Each received frame is processed by each state machine (unless discarded) in the order listed below:

- a) Edge (see 7.6.3.6): Frames received across an edge are stripped.
- b) Check (see 7.6.3.7): Errored frames are discarded.
- c) Count (see 7.6.3.8): Traffic-flow statistics are updated.
- d) Strip (see 7.6.3.9): Selected frames are stripped or transited.
- e) Adjust (see 7.6.3.10): Frames headers are adjusted.
- f) Queue (see 7.6.3.10): Frames are queued into PTQ or STQ.
- g) Filter (see 7.6.3.3): Frames destined to the client are filtered.
- h) Rx SAS (see 7.6.3.5): Frames destined to the client are processed by SAS.

7.6.3.2 Strip rules overview

7.6.3.3 Filter entity

Replace the content of 7.6.3.3 with the following.

Each filter entity is composed of select and count subentities, as illustrated in Figure 7.9, which use rules designed to support local and bridged RPR traffic.

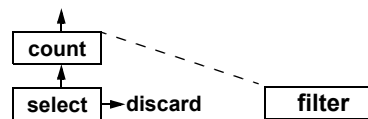


Figure 7.9—Filter subentities

The filter entity includes the following subentities, detailed in 7.6.3.12–7.6.3.13. Each received frame is processed by each state machine (unless discarded) in the order listed below:

- a) Select (see 7.6.3.12): Frames are selected for one of the following actions:
 - 1) Data: data frames are passed to the client.
 - 2) Control: control frames are passed to the MAC control entity.
 - 3) Discard: invalid data and control frames are discarded.
- b) Count (see 7.6.3.12 and 7.6.3.13): Selected frames are counted as follows:
 - 1) Data: data frames and bytes are counted per service class before being passed to the client.
 - 2) Control: control frames are counted per *controlType* before being passed to the client.

7.6.3.4 Filter rules overview

Insert 7.6.3.5 after 7.6.3.4 and renumber subsequent subclauses accordingly. All references in the new text refer to the new numbering.

7.6.3.5 Receive SAS entity

Each receive SAS entity is composed of learn and map subentities, as illustrated in Figure 7.10, which use rules designed to support local and bridged RPR traffic.

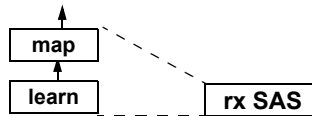


Figure 7.10—Rx SAS subentities

The receive SAS entity includes the following subentities, detailed in 14.5.4. Each received frame is processed by each state machine in the order listed below:

- Learn (see 14.5.4.1): Learns associations between station address and remote addresses, for SAS frames.
- Map (see 14.5.4.1): Optionally remaps extended addresses to base addresses.

7.6.3.12 ReceiveFilterDataCount state machine

7.6.3.12.4 ReceiveFilterDataCount state table

Replace contents of 7.6.3.12.4 with the following.

The ReceiveFilterDataCount state machine specified in Table 7.25 implements the functions necessary for counting bytes and frames to be copied to the client. In the case of any ambiguity between the text and the state machine, the state machine shall take precedence. The notation used in the state table is described in 3.4.

Table 7.25—ReceiveFilterDataCount state table

Current state		Row	Next state	
state	condition		action	state
START	(frame = Dequeue(Q_RX_F_D_CNT)) != NULL	1	size = SizeOf(frame);	UCAST
	—	2	—	START

Table 7.25—ReceiveFilterDataCount state table (continued)

Current state		Row	Next state	
state	condition		action	state
UCAST	Broadcast(frame.da)	3	toClientBcastFrames += 1;	MCAST
	Multicast(frame.da)	4	toClientMcastFrames += 1;	
	frame.sc == CLASS_A0	5	toClientUcastClassAFrames += 1;	FINAL
	frame.sc == CLASS_A1	6	toClientUcastClassABytes += size;	
	frame.sc == CLASS_B && frame.fe == 0	7	toClientUcastClassBCirFrames += 1; toClientUcastClassBCirBytes += size;	
	frame.sc == CLASS_B && frame.fe == 1	8	toClientUcastClassBEirFrames += 1; toClientUcastClassBEirBytes += size;	
	frame.sc == CLASS_C	9	toClientUcastClassCFrames += 1; toClientUcastClassCBytes += size;	
MCAST	frame.sc == CLASS_A0	10	toClientMcastClassAFrames += 1;	FINAL
	frame.sc == CLASS_A1	11	toClientMcastClassABytes += size;	
	frame.sc == CLASS_B && frame.fe == 0	12	toClientMcastClassBCirFrames += 1; toClientMcastClassBCirBytes += size;	
	frame.sc == CLASS_B && frame.fe == 1	13	toClientMcastClassBEirFrames += 1; toClientMcastClassBEirBytes += size;	
	frame.sc == CLASS_C	14	toClientMcastClassCFrames += 1; toClientMcastClassCBytes += size;	
FINAL	—	15	Enqueue(Q_RX_SAS, frame);	START

Row 7.25-1: Fetch a new input frame.

Row 7.25-2: Refetch an input frame, after each NULL value is returned.

Row 7.25-3: Broadcast frames have distinct counters.

Row 7.25-4: Multicast frames have distinct counters.

Row 7.25-5: Count the number of unicast classA frames and bytes, subclassA0 contribution.

Row 7.25-6: Count the number of unicast classA frames and bytes, subclassA1 contribution.

Row 7.25-7: Count the number of unicast classB-CIR frames and bytes.

Row 7.25-8: Count the number of unicast classB-EIR frames and bytes.

Row 7.25-9: Count the number of unicast classC frames and bytes.

Row 7.25-10: Count the number of multicast classA frames and bytes, subclassA0 contribution.

Row 7.25-11: Count the number of multicast classA frames and bytes, subclassA1 contribution.

Row 7.25-12: Count the number of multicast classB-CIR frames and bytes.

Row 7.25-13: Count the number of multicast classB-EIR frames and bytes.

Row 7.25-14: Count the number of multicast classC frames and bytes.

Row 7.25-15: Pass the frames upward toward the client, through SAS.

7.6.3.13 ReceiveFilterControlCount state machine

7.6.3.13.1 ReceiveFilterControlCount state machine definitions

Insert the following definitions in alphabetical order into 7.6.3.13.1.

CT_OAM_SAS_NOTIFY
Q_RX_SAS_NOTIFY

7.6.3.13.2 ReceiveFilterControlCount state machine variables

Insert the following variables in alphabetical order in 7.6.3.13.2.

toCtrlOamSasNotifyFrames

7.6.3.13.4 ReceiveFilterControlCount state table

Replace the content of 7.6.3.13.4 with the following.

The ReceiveFilterControlCount state machine specified in Table 7.26 implements the functions necessary for counting control frames to be copied to the MAC control sublayer. In the case of any ambiguity between the text and the state machine, the state machine shall take precedence. The notation used in the state table is described in 3.4.

Table 7.26—ReceiveFilterControlCount state table

Current state		Row	Next state	
state	condition		action	state
START	(frame = Dequeue(Q_RX_F_C_CNT)) != NULL	1	—	COUNT
	—	2	—	START

Table 7.26—ReceiveFilterControlCount state table (continued)

Current state		Row	Next state	
state	condition		action	state
COUNT	frame.ft == FT_FAIRNESS	3	toCtrlFairnessFrames += 1; Enqueue(Q_RX_FAIR, frame);	FINAL
	frame.controlType == CT_STATION_ATD	4	toCtrlTopoATDFrames += 1; Enqueue(Q_RX_ATD, frame);	
	frame.controlType == CT_TOPO_PROT	5	toCtrlTopoTPFrames += 1; Enqueue(Q_RX_TP, frame);	
	frame.controlType == CT_TOPO_CHKSUM	6	toCtrlTopoSumFrames += 1; Enqueue(Q_RX_TC, frame);	
	frame.controlType == CT_LRTT_REQ	7	toCtrlLrttReqFrames += 1; Enqueue(Q_RX_LRTT_REQ, frame);	
	frame.controlType == CT_LRTT_RSP	8	toCtrlLrttRspFrames += 1; Enqueue(Q_RX_LRTT_RSP, frame);	
	frame.controlType == CT_FDD	9	toCtrlFddFrames += 1; Enqueue(Q_RX_FDD, frame);	
	frame.controlType == CT_OAM_ECHO_REQ	10	toCtrlOamEchoReqFrames += 1; Enqueue(Q_RX_ECHO_REQ, frame);	
	frame.controlType == CT_OAM_ECHO_RSP	11	toCtrlOamEchoRspFrames += 1; Enqueue(Q_RX_ECHO_RSP, frame);	
	frame.controlType == CT_OAM_FLUSH	12	toCtrlOamFlushFrames += 1; Enqueue(Q_RX_FLUSH, frame);	
	frame.controlType == CT_OAM_ORG	13	toCtrlOamOrgFrames += 1; Enqueue(Q_RX_ORG, frame);	
	frame.controlType == CT_OAM_SAS_NOTIFY	14	toCtrlOamSasNotifyFrames += 1; Enqueue(Q_RX_SAS_NOTIFY, frame);	
	—	15	badControlTypeFrames += 1;	START
FINAL	—	16	toCtrlFrames += 1;	START

Row 7.2-1: Fetch a new input frame.**Row 7.2-2:** Refetch an input frame, after each NULL value is returned.**Row 7.2-3:** Count the number of fairness frames.**Row 7.26-4:** Count the number of station attribute discovery frames.**Row 7.26-5:** Count the number of topology and protection frames.**Row 7.26-6:** Count the number of topology checksum frames.**Row 7.26-7:** Count the number of loop round trip time request frames.**Row 7.26-8:** Count the number of loop round trip time response frames.**Row 7.26-9:** Count the number of fairness differential delay frames.**Row 7.26-10:** Count the number of OAM echo request frames.**Row 7.26-11:** Count the number of OAM echo response frames.**Row 7.26-12:** Count the number of OAM flush frames.**Row 7.26-13:** Count the number of OAM organization specific frames.

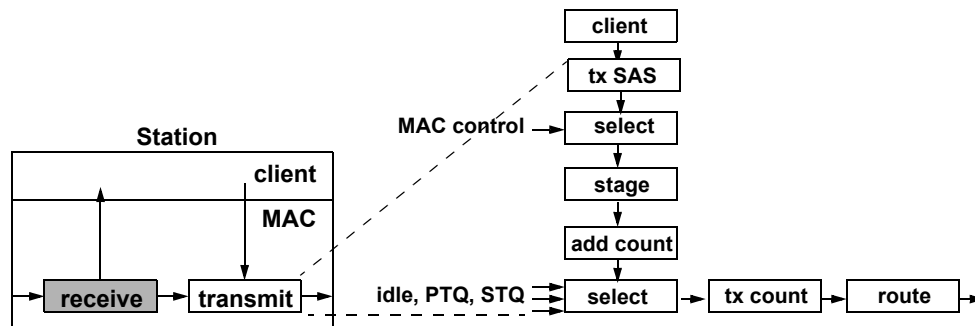
Row 7.26-14: Count the number of SAS notify frames.

Row 7.26-15: Don't count any other frames.

Row 7.26-16: Count all the frames in aggregate and pass them to the MAC control sublayer.

7.7 Transmit operation

Replace Figure 7.10 with the following figure.



Insert the following subentity description at the beginning of the list. Renumber the list as necessary.

- a) spatially aware sublayer (see 14.5.3)

7.7.1 Ringlet selection

7.7.1.6 RingletSelection state machine

7.7.1.6.2 RingletSelection state machine variables

Remove the following variable (and accompanying reference) from the list.

forceStrict
See 7.2.4.

7.7.1.6.3 RingletSelection state machine routines

Insert the following routines in alphabetical order into 7.7.1.6.3.

CheckEnqueueRi(ringlet, queue, frame)
If frame.ttl is greater than 0, then call *EnqueueRi(ringlet, queue, frame)*.
EnqueueRi(ringlet, queue, frame)
See 7.2.3.

Replace the definition of the following routines, while maintaining alphabetical order, with the following.

InitialTtl(frame)
Returns the initial value for the *ttl* field of *frame* given the values of *da* and *ri* within the constraints detailed in 7.7.3, including optionally searching the SDB static group address table for a matching static entry.
Provided(parameter)
See 7.2.3.

Reachable(address, ringlet)
See 7.2.3.

7.7.1.6.4 RingletSelection state table

Replace Table 7.29 with the table below.

Current state		Row	Next state	
state	condition		action	state
START	(frame = Dequeue(Q_TX_RS)) == NULL	1	—	START
	ringlet_id == RI_0 ringlet_id == RI_1	2	—	PASSED
	—	3	ringlet_id = DefaultRinglet();	
PASSED	—	4	frame.ps = 0;	STRICT
STRICT	frame.ft == FT_CONTROL	5	frame.da = destination_address; frame.sa = myMacAddress; frame.ri = ringlet_id; frame.we = 0; frame.ef = 0; frame.fi = FI_NONE; frame.so = 0;	TRANSMIT
	containmentActive && (transmissionControl == STRICT)	6	containedFrames += 1;	START
	transmissionControl == STRICT	7	frame.so = 1;	PROTECT
	containmentActive && (!Provided(strict_order) strict_order)	8	containedFrames += 1;	START
	!Provided(strict_order) strict_order	9	frame.so = 1;	PROTECT
	—	10	frame.so = 0;	
PROTECT	!mac_protection	11	frame.ri = ringlet_id; frame.we = 0;	EXTEND
	frame.so && protConfig == WRAPPING && EdgeOfRing() && myProtectMethod == CENTER_WRAP	12	frame.ri = ringlet_id; frame.we = 1; frame.ps = (frame.ri == MyEdgeRinglet());	
	protConfig == WRAPPING && EdgeOfRing() && myProtectMethod == CENTER_WRAP	13	frame.ri = Other(MyEdgeRinglet()); frame.we = 1;	
	protConfig == WRAPPING	14	frame.ri = ringlet_id; frame.we = 1;	
	—	15	frame.ri = ringlet_id; frame.we = 0;	

Current state		Row	Next state	
state	condition		action	state
EXTEND	Provided(source_address_extended)	16	Extend(frame); frame.sa = myMacAddress; frame.da = destination_address; frame.saExtended = source_address_extended; frame.daExtended = destination_address_extended; frame.ef = 1;	DA
	Provided(source_address) && source_address != myMacAddress	17	Extend(frame); frame.sa = myMacAddress; frame.da = destination_address; frame.saExtended = source_address; frame.daExtended = destination_address; frame.ef = 1;	
	—	18	frame.sa = myMacAddress; frame.da = destination_address; frame.ef = 0;	
DA	optionSecMacSubstitute && PriAddress(destination_address) != NULL	19	frame.da = PriAddress (destination_address);	FLOOD
	—	20	—	
FLOOD	Provided(flooding_form) && flooding_form == FLOOD_NONE	21	frame.fi = FI_NONE;	TRANSMIT
	Provided(flooding_form) && flooding_form != FLOOD_NONE	22	—	FORM
	Provided(source_address) && source_address != myMacAddress	23	—	
	Unicast(frame.da) && Reachable(frame.da, ringlet_id)	24	frame.fi = FI_NONE;	TRANSMIT
	Unicast(frame.da) && frame.we == 1 && Reachable(frame.da, Other(ringlet_id))	25	frame.fi = FI_NONE;	
	Unicast(frame.da) && mac_protection && Reachable(frame.da, Other(ringlet_id))	26	frame.ri = Other(ringlet_id); frame.fi = FI_NONE;	
	Unicast(frame.da) && !mac_protection && Reachable(frame.da, Other(ringlet_id))	27	frame.fi = FI_NONE;	
	—	28	—	FORM
FORM	(Provided(flooding_form) && flooding_form == FLOOD_BIDIR) (mac_protection && protConfig == STEERING && topoType == OPEN_RING) (!Provided(flooding_form) && myFloodingForm == FLOOD_BIDIR)	29	frame.fi = FI_BIDIR;	BIEDGE
	—	30	frame.fi = FI_UNIDIR;	TRANSMIT

Current state		Row	Next state	
state	condition		action	state
BIEDGE	IsEdge(RINGLET_0) && IsEdge(RINGLET_1)	31	—	START
	IsEdge(RINGLET_0)	32	frame.ri = RINGLET_1;	TRANSMIT
	IsEdge(RINGLET_1)	33	frame.ri = RINGLET_0;	
	—	34	Replicate(frame); frame0.ri = RINGLET_0; frame0.fi = FI_BIDIR; frame0.ttl = InitialTtl(frame0); frame0.ttlBase = frame0.ttl; frame0.hec = Crc16(frame0); frame0.fcs = Crc32(frame0); frame1.ri = RINGLET_1; frame1.fi = FI_BIDIR; frame1.ttl = InitialTtl(frame1); frame1.ttlBase = frame1.ttl; frame1.hec = Crc16(frame1); frame1.fcs = Crc32(frame1); CheckEnqueueRi(RINGLET_0, Q_TX_SS, frame0); CheckEnqueueRi(RINGLET_1, Q_TX_SS, frame1);	START
TRANSMIT	IsEdge(frame.ri) && (frame.we == 0 protConfig == STEERING) && !EdgeAllowedFrame(frame)	35	—	START
	protConfig == WRAPPING && EdgeOfRing() && myProtectMethod == CENTER_WRAP	36	frame.ttl = InitialTtl(frame); frame.ttlBase = frame.ttl; frame.hec = Crc16(frame); frame.fcs = Crc32(frame); CheckEnqueueRi(MyEdgeRinglet(), Q_TX_SS, frame);	
	—	37	frame.ttl = InitialTtl(frame); frame.ttlBase = frame.ttl; frame.hec = Crc16(frame); frame.fcs = Crc32(frame); CheckEnqueueRi(frame.ri, Q_TX_SS, frame);	

7.7.2 Determination of cleave point

Change the second paragraph of 7.7.2 as follows.

In the case of a ring that is currently experiencing a protection event, the cleave point for replicated frames is the point at which the protection event exists, regardless of the value of *stdCleave* (see 6.2.4). If more than one protection event exists, a separate cleave point is chosen for each ringlet, with the cleave point being the first span known to be an edge in the transmit direction of the ringlet.

Replace the third paragraph with the following paragraph.

For non-SAS stations, the determination of the cleave point for replicated frames is implementation specific when the ring is not currently experiencing a protection event. The cleave point should not change unless the topology changes (either topology type or contents of topology), to avoid the possibility of reordering frames apart from the cleave point selection.

Prior to 7.7.3, insert 7.7.2.1 with the following content.

7.7.2.1 Cleave point determination for SAS

To facilitate interoperability of SAS implementations, a station may use a standardized SAS cleave point calculation. This calculation is controlled by the value of *stdCleave* (see 6.2.4).

For rings with an odd number of stations, the cleave point is selected such that an equal number of stations appear to the east and west of the sourcing station, as illustrated in Figure 7.12.

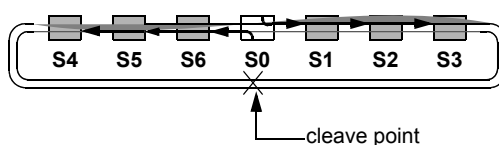
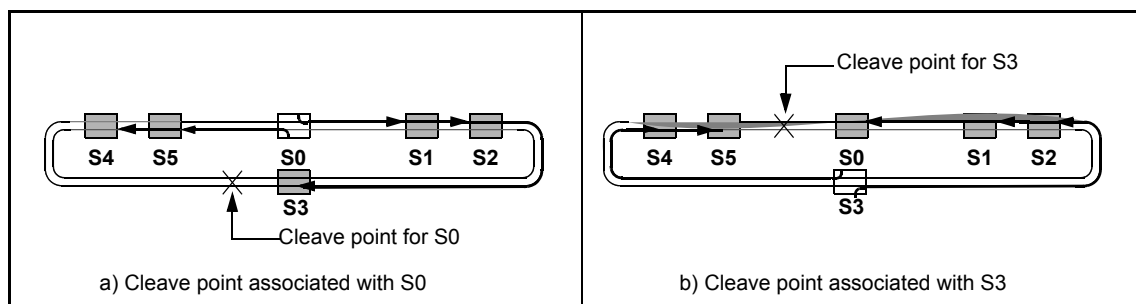


Figure 7.12—Odd number of stations cleave point example

For rings with an even number of stations, cleave points are selected such that diametrically opposite stations send flooded traffic to each other over the same path, as illustrated in Figure 7.13. The station with the lower MAC address sets its cleave point to the east of its opposite station, and the station with the higher MAC address sets its cleave point to the west of its opposite station.



NOTE—S0 MAC address is lower than S3 MAC address.

Figure 7.13—Even number of stations cleave point example

7.7.3 Setting of *ttl* and *ttlBase*

7.7.3.1 Definitions for use with setting of *ttl* and *ttlBase*

Insert the following definition in alphabetical order.

CT_OAM_SAS_NOTIFY

7.7.3.2 Variables for use with setting of *ttl* and *ttlBase*

Insert the following variable and description in alphabetical order.

optionSasMcastScope

See 7.2.4.

7.7.3.3 Routines for use with setting of *ttl* and *ttlBase*

Insert the following routines and description in alphabetical order.

SasMcastScope(frame)

Returns TRUE, if the *optionSasMcastScope* variable is set to TRUE and the frame.*da* is a group address and the *SdbMcastLookup*() routine returns a value other than NULL. Returns FALSE otherwise.

SdbMcastLookup(frame)

If the frame is extended, then the extended destination address of the frame (and VLAN identifier (VID) if *optionSasVlanAware* is TRUE) is used as the index to the SDB static multicast lookup table. If the frame is not extended, then the destination address of the frame (and VID if *optionSasVlanAware* is TRUE) is used as the index to the SDB static multicast lookup table. If an entry is found (either destination address only, or destination address and VID), the *sasTxMcastScoped-Frames* counter is incremented and a valid hop count associated with frame.*ri* is returned. A hop count is valid only if $(sdbStaticMcastHopsRinglet0 + sdbStaticMcastHopsRinglet1) < numStations$. Otherwise, if an entry is not found or a valid hop count cannot be returned, a value of NULL is returned.

Unicast(macAddress)

See 7.2.3.

7.7.3.4 Initial *ttl* field values

Replace the contents of 7.7.3.4 with the following.

Every frame transmitted from the local client or local MAC control sublayer shall have its *ttl* field set according to the constraint rules specified in Table 7.30, where the rows are evaluated in top-to-bottom order. When a given constraint is a range of values, any value within the range is allowed.

Table 7.30—Initial *ttl* field values

ft field value	Additional relevant values	Row	Constraints on initial <i>ttl</i> field value
FT_IDLE	—	1	$ttl == 1$

Table 7.30—Initial *ttl* field values (continued)

ft field value	Additional relevant values	Row	Constraints on initial <i>ttl</i> field value
FT_CONTROL	frame.controlType == CT_STATION_ATD	2	ttl == MAX_STATIONS
	frame.controlType == CT_TOPO_CHKSUM	3	ttl == 1
	frame.controlType == CT_TOPO_PROT	4	ttl == MAX_STATIONS
	frame.controlType == CT_LRRTT_REQ	5	ttl >= NumberOfHops(frame.da) && ttl <= MAX_STATIONS
	frame.controlType == CT_LRRTT_RSP	6	ttl >= NumberOfHops(frame.da) && ttl <= MAX_STATIONS
	frame.controlType == CT_FDD	7	ttl >= NumberOfHops(frame.da) && ttl <= MAX_STATIONS
	frame.controlType == CT_OAM_ECHO_REQ	8	ttl >= NumberOfHops(frame.da) && ttl <= MAX_STATIONS
	frame.controlType == CT_OAM_ECHO_RSP	9	ttl >= NumberOfHops(frame.da) && ttl <= MAX_STATIONS
	frame.controlType == CT_OAM_FLUSH	10	ttl >= numStations - 1 && ttl <= MAX_STATIONS
	frame.controlType == CT_OAM_ORG	11	ttl >= NumberOfHops(frame.da) && ttl <= MAX_STATIONS
	frame.controlType == CT_OAM_SAS_NOTIFY	12	ttl >= numStations - 1 && ttl <= MAX_STATIONS
FT_FAIRNESS	frame.ffType == SINGLE_CHOKE	13	ttl == MAX_STATIONS
	frame.ffType == MULTI_CHOKE	14	ttl == MAX_STATIONS
FT_DATA	SasMcastScope(frame)	15	ttl == SdbMcastLookup(frame)
	!SasMcastScope(frame) && (frame.fi == FI_BIDIR)	16	ttl == NumberOfHops(cleavePoint)
	!SasMcastScope(frame) && (frame.fi == FI_UNIDIR)	17	ttl >= numStations - 1 && ttl <= MAX_STATIONS
	!SasMcastScope(frame) && (frame.fi == FI_NONE)	18	ttl >= NumberOfHops(frame.da) && ttl <= MAX_STATIONS

Row 7.30-1: Idle frames always travel one hop.

Row 7.30-2: Station attribute discovery frames are sent without (accurate) knowledge of the size of the ring and are capable of traveling the largest possible ring.

Row 7.30-3: Topology checksum frames always travel one hop.

Row 7.30-4: Topology and protection frames are sent without (accurate) knowledge of the size of the ring and are capable of traveling the largest possible ring.

Row 7.30-5: Loop round trip time request frames are capable of traveling as far as their intended destination.

Row 7.30-6: Loop round trip time response frames are capable of traveling as far as their intended destination.

Row 7.30-7: Fairness differential delay frames are capable of traveling as far as their intended destination.

Row 7.30-8: Echo request frames are capable of traveling as far as their intended destination.

Row 7.30-9: Echo response frames are capable of traveling as far as their intended destination.

Row 7.30-10: Flush frames are capable of traveling the known size of the ring.

Row 7.30-11: Organization specific frames are capable of traveling as far as their intended destination.

Row 7.30-12: SAS notify frames are capable of traveling the known size of the ring.

Row 7.30-13: Single-choke fairness frames always travel one hop, but start with a *ttl* of MAX_STATIONS whenever the *saCompact* value is reset to the local station. Otherwise, the *ttl* is decremented from the last received *ttl* when the frame is regenerated by the local station.

Row 7.30-14: Multi-choke fairness frames are sent without (accurate) knowledge of the size of the ring and are capable of traveling the largest possible ring.

Row 7.30-15: If the SAS multicast scoping feature is active, and a static multicast entry is found in the *SdbMcastLookup* table, then *ttl* is set to the result of the table lookup. If the entry found specifies 0 hops (i.e., don't send the frame), then the frame will be discarded.

Row 7.30-16: Bidirectionally flooded frames travel exactly as far as the cleave point, such that the following condition is satisfied: $frame0.ttl + frame1.ttl == numStations - 1$.

Row 7.30-17: Unidirectionally flooded frames are capable of traveling the known size of the ring.

Row 7.30-18: Frames that are not flooded travel at least as far as their intended destination.

NOTE—Setting *ttl* to a value larger than needed can interfere with bandwidth reclamation, which impedes full ring bandwidth utilization.

7.7.6 ControlAddCount state machine

7.7.6.1 ControlAddCount state machine definitions

Insert the following definition in alphabetical order in 7.7.6.1.

CT_OAM_SAS_NOTIFY

7.7.6.2 ControlAddCount state machine variables

Insert the following variables in alphabetical order in 7.7.6.2.

fromCtrlOamSasNotifyFrames

7.7.6.4 ControlAddCount state table

Replace the contents of 7.7.6.4 with the following.

The ControlAddCount state machine counts added control frames, as specified in Table 7.33. In the case of any ambiguity between the text and the state machine, the state machine shall take precedence. The notation used in the state table is described in 3.4.

Table 7.33—ControlAddCount state table

Current state		Row	Next state	
state	condition		action	state
START	(frame = Dequeue(Q_TX_C_COUNT)) != NULL	1	—	COUNT
	—	2	—	START

Table 7.33—ControlAddCount state table (continued)

Current state		Row	Next state	
state	condition		action	state
COUNT	frame.controlType == CT_STATION_ATD	3	fromCtrlTopoATDFrames += 1;	STAGE
	frame.controlType == CT_TOPO_PROT	4	fromCtrlTopoTPFrames += 1;	
	frame.controlType == CT_TOPO_CHKSUM	5	fromCtrlTopoSumFrames += 1;	
	frame.controlType == CT_LRTT_REQ	6	fromCtrlLrttReqFrames += 1;	
	frame.controlType == CT_LRTT_RSP	7	fromCtrlLrttRspFrames += 1;	
	frame.controlType == CT_FDD	8	fromCtrlFddFrames += 1;	
	frame.controlType == CT_OAM_ECHO_REQ	9	fromCtrlOamEchoReqFrames += 1;	
	frame.controlType == CT_OAM_ECHO_RSP	10	fromCtrlOamEchoRspFrames += 1;	
	frame.controlType == CT_OAM_FLUSH	11	fromCtrlOamFlushFrames += 1;	
	frame.controlType == CT_OAM_ORG	12	fromCtrlOamOrgFrames += 1;	
	frame.controlType == CT_OAM_SAS_NOTIFY	13	fromCtrlOamSasNotifyFrames += 1;	
	—	14	—	
STAGE	—	15	fromCtrlFrames += 1; Enqueue(Q_TX_STAGE, frame);	START

Row 7.33-1: Fetch the next add frame.

Row 7.33-2: Wait for an add frame.

Row 7.33-3: Count the number of station attribute discovery frames.

Row 7.33-4: Count the number of topology and protection frames.

Row 7.33-5: Count the number of topology checksum frames.

Row 7.33-6: Count the number of loop round trip time request frames.

Row 7.33-7: Count the number of loop round trip time response frames.

Row 7.33-8: Count the number of fairness differential delay frames.

Row 7.33-9: Count the number of OAM echo request frames.

Row 7.33-10: Count the number of OAM echo response frames.

Row 7.33-11: Count the number of OAM flush frames.

Row 7.33-12: Count the number of OAM organization specific frames.

Row 7.33-13: Count the number of SAS notify frames.

Row 7.33-14: Don't count any other frames.

Row 7.33-15: Count all frames and add them to the tail of the stage queue.

7.8 Protocol Implementation Conformance Statement (PICS) proforma for Clause 7²

7.8.3 PICS tables for Clause 7

7.8.3.7 *t*tl setting

Insert the following entry, at the end the table.

TTL3	<i>stdCleave</i> setting	7.7.2.1	Standard cleave point setting is supported	O	Yes [] No []
------	--------------------------	---------	--	---	-------------------

²*Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this clause so that it can be used for its intended purpose and may further publish the completed PICS.

9. Frame formats

9.3 Control frame format

9.3.2 Control frame fields

Insert the following value 0B₁₆ information field in Table 9.4, after the 0A₁₆ entry.

0B ₁₆	CT_OAM_SAS_NOTIFY	OAM SAS notify frame
------------------	-------------------	----------------------

