INTERNATIONAL STANDARD

# ISO/IEC/IEEE 8802-1AX

Second edition
2021-09



# Telecommunications and exchange between information technology systems — Requirements for local and metropolitan area networks —

## Part 1AX: Link aggregation

*Télécommunications et échange entre systèmes informatiques — Exigences pour les réseaux locaux et métropolitains —*

*Partie 1AX: Agrégation de lien*

Reference number
ISO/IEC/IEEE 8802-1AX:2021(E)



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

Email: stds.ipr@ieee.org
Website: www.ieee.org

Published in Switzerland

# Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO/IEC documents should be noted.

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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

ISO/IEC/IEEE 8802-1AX was prepared by the LAN/MAN of the IEEE Computer Society (as IEEE Std 802.1AX-2020) and drafted in accordance with its editorial rules. It was adopted, under the "fast-track procedure", defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*.

This second edition cancels and replaces the first edition (ISO/IEC/IEEE 8802-1AX:2016), which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC/IEEE 8802-1AX:2016/Cor 1:2018.

A list of all parts in the ISO/IEC/IEEE 8802 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.



**IEEE Std 802.1AX™-2020**
(Revision of IEEE Std 802.1AX-2014)

# IEEE Standard for Local and Metropolitan Area Networks—

# Link Aggregation

Developed by the

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

Approved 30 January 2020

**IEEE SA Standards Board**

**Abstract:** Link Aggregation allows parallel point-to-point links to be used as if they were a single link and also supports the use of multiple links as a resilient load-sharing interconnect between multiple nodes in two separately administered networks. This standard defines a MAC-independent Link Aggregation capability and provides general information relevant to specific MAC types.

**Keywords:** Aggregated Link, Aggregator, Distributed Resilient Network Interconnect, DRNI, interconnect, Link Aggregation, Link Aggregation Group, local area network, management, Network-Network Interface, NNI



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



PDF: ISBN 978-1-5044-6428-4 STD24045
Print: ISBN 978-1-5044-6429-1 STDPD24045

# Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at https://standards.ieee.org/ipr/disclaimers.html.

## Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. 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.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

## Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

## Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. 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 of IEEE.

## Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive 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 comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

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

## Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

## Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

## Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, 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.

## Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, 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 order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore at https://ieeexplore.ieee.org/ or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website at https://standards.ieee.org.

## Errata

Errata, if any, for IEEE standards can be accessed via https://standards.ieee.org/standard/index.html. Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore: https://ieeexplore.ieee.org/browse/standards/collection/ieee/. Users are encouraged to periodically check for errata.

## 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 by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at https://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

## Participants

At the time this revision was submitted to the IEEE SA for approval, the IEEE P802.1 Working Group had the following membership:

**Glenn Parsons,** *Chair*
**John Messenger,** *Vice Chair*
**János Farkas**, *Chair, Time-Sensitive Networking Task Group*
**Stephen Haddock** and **Jessy Rouyer**, *Co-Editors*

Astrit Ademaj
Ralf Assmann
Jens Bierschenk
Christian Boiger
Paul Bottorff
Radhakrishna Canchi
Feng Chen
Weiying Cheng
Paul Congdon
Rodney Cummings
Josef Dorr
Hesham Elbakoury
Thomas Enzinger
Donald Fedyk
Norman Finn
Geoffrey Garner
Craig Gunther
Marina Gutierrez
Mark Hantel
Marc Holness
Satoko Itaya
Yoshihiro Ito
Michael Karl
Stephan Kehrer
Randy Kelsey
Hajime Koto
James Lawlis
Christophe Mangin
Scott Mansfield
Kenichi Maruhashi
David McCall
Larry McMillan
Tero Mustala
Roy Myers
Hiroki Nakano
Bob Noseworthy
Tomoki Ohsawa
Hiroshi Ohue
Donald R. Pannell
Michael Potts
Dieter Proell
Wei Qiu
Karen Randall
Maximilian Riegel
Atsushi Sato
Frank Schewe
Michael Seaman
Maik Seewald
Johannes Specht
Marius Stanica
Guenter Steindl
Karim Traore
Hao Wang
Tongtong Wang
Xinyuan Wang
Karl Weber
Brian Weis
Ludwig Winkel
Jordon Woods
Takahiro Yamaura
Nader Zein
William Zhao
Helge Zinner
Harald Zweck

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

Thomas Alexander
Richard Alfvin
Butch Anton
Stefan Aust
Harry Bims
Kenneth Bow
Rich Boyer
Nancy Bravin
Vern Brethour
Demetrio Bucaneg
William Byrd
Radhakrishna Canchi
Paul Cardinal
Juan Carreon
Pin Chang
János Farkas
John Fletcher
Avraham Freedman
Devon Gayle
Zhigang Gong
Randall Groves
Stephen Haddock
Marek Hajduczenia
Marco Hernandez
Werner Hoelzl
Yasuhiro Hyakutake
Satoko Itaya
Raj Jain
Sangkwon Jeong
Pranav Jha
Lokesh Kabra
Srinivas Kandala
Piotr Karocki
Stuart Kerry
Evgeny Khorov
Yongbum Kim
Robert Landman
Hyeong Ho Lee
John Lemon
James Lepp
Michael Lynch
John Mackay
Roger Marks
Arthur Marris
Jeffery Masters
Brett Mcclellan
Richard Mellitz
Jose Morales
Ronald Murias
Bruce Muschlitz
Nick S. A. Nikjoo
Satoshi Obara
Carlos Pardo
Bansi Patel
Brian Petry
David Piehler
Walter Pienciak
Rick Pimpinella
Clinton Powell
R. K. Rannow
Maximilian Riegel
Robert Robinson
Jessy Rouyer
John Sargent
Frank Schewe
Michael Seaman
Thomas Starai
Walter Struppler
Michael Thompson
Mark-Rene Uchida
James Van De Ligt
Dmitri Varsanofiev
George Vlantis
Stephen Webb
Karl Weber
Hung-Yu Wei
Scott Willy
Peter Wu
Oren Yuen
Nader Zein

When the IEEE-SA Standards Board approved this standard on 30 January 2020, it had the following membership:

**Gary Hoffman,** Chair
**Vacant Position,** Vice Chair
**Jean-Philippe Faure,** Past Chair
**Konstantinos Karachalios,** Secretary

Ted Burse
J. Travis Griffith
Grace Gu
Guido R. Hiertz
Joseph L. Koepfinger*
John D. Kulick
David J. Law
Howard Li
Dong Liu
Kevin Lu
Paul Nikolich
Damir Novosel
Jon Walter Rosdahl
Dorothy Stanley
Mehmet Ulema
Lei Wang
Sha Wei
Philip B. Winston
Daidi Zhong
Jingyi Zhou

*Member Emeritus

# Introduction

(This introduction is not part of IEEE Std 802.1AX-2020, IEEE Standard for Local and Metropolitan Area Networks—Link Aggregation.)

Link Aggregation allows one or more links to be aggregated together to form a Link Aggregation Group (LAG) so that the Link Aggregation Client can treat the LAG as if it were a single link. Link Aggregation was originally published as IEEE Std 802.3ad™-2000 and subsequently incorporated into the IEEE Std 802.3™, 2000 Edition. In 2008 Link Aggregation was removed from IEEE Std 802.3 and published as IEEE Std 802.1AX-2008. These standards specified the aggregation of full-duplex point-to-point links using IEEE Std 802.3 media of the same speed.

An amendment, IEEE Std 802.1AXbk™-2012, specified changes to the addressing used by the link aggregation control and marker protocols to allow a LAG to span Two-Port Media Access Control (MAC) Relays (TPMRs) and to span Provider Bridged Networks and Provider Backbone Bridge Networks.

A revision, IEEE Std 802.1AX-2014, extended Link Aggregation in three areas. First, it explicitly allowed the aggregation of point-to-point links of any speed using any physical media or logical connection capable of supporting the Internal Sublayer Service specified in IEEE Std 802.1AC™. Second, it specified Conversation-Sensitive Collection and Distribution (CSCD) that provides a mechanism to identify the distribution algorithm in use to map data frames to individual links in the LAG and to convey that information to the Link Aggregation Partner via Link Aggregation Control Protocol Data Units (LACPDUs) containing version 2 type/length/values (TLVs). Third, it specified Distributed Resilient Network Interconnect (DRNI) that allows a LAG to terminate at two or three cooperating Systems so that the LAG provides resiliency to System-level failures as well as link level failures. A corrigendum, IEEE Std 802.1AX-2014/Cor 1-2017, provided technical and editorial corrections to CSCD.

This revision, IEEE Std 802.1AX-2020, makes significant refinements and simplifications to the Link Aggregation Control Protocol (LACP) as well as to CSCD and DRNI. In LACP, the Periodic state machine and Transmit state machine are combined to a single machine, and the Mux machine is optimized to reduce the likelihood of excessive Link Aggregation Control Protocol Data Unit (LACPDU) transmissions. CSCD is refined to eliminate the TLVs that led to LACPDUs greater than 128 bytes in length. DRNI is significantly revised and simplified to support a LAG terminating at just two (not three) cooperating Systems.

Every effort has been made to maintain interoperability, without prior configuration, with LACP implementations conforming to IEEE Std 802.3ad-2000, IEEE Std 802.1AX-2008, or IEEE Std 802.1AX-2014 and with CSCD implementations conforming to IEEE Std 802.1AX-2014. The changes to DRNI, and in particular the Distributed Relay Control Protocol (DRCP), are such that an implementation conforming to this standard will not interoperate with a DRCP implementation conforming to IEEE Std 802.1AX-2014. The DRCP version number in this standard has been changed to version 2, and care has been taken so that a DRCP implementation conformant to IEEE Std 802.1AX-2014 will discard version 2 DRCPDUs as invalid and that implementations of this standard will discard version 1 DRCPDUs.

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material. Information on the current revision state of this and other IEEE 802 standards may be obtained from

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

# Contents

# Figures

## Tables

# IEEE Standard for Local and Metropolitan Area Networks—

# Link Aggregation



## 1. Overview

### 1.1 Scope

Link Aggregation provides protocols, procedures, and managed objects that allow the following:

— One or more parallel instances of point-to-point links to be aggregated together to form a Link Aggregation Group (LAG) so that a Link Aggregation Client can treat the LAG as if it were a single link.

— Conversation-Sensitive Collection and Distribution (CSCD) that specifies a means to identify the distribution algorithm in use to assign frames to individual links in a LAG and to convey that information to the System at the other end of the LAG.

— Distributed Resilient Network Interface (DRNI) that enables a LAG to terminate at a pair of cooperating Systems in order to provide system-level as well as link-level resiliency.

### 1.2 Purpose

Link Aggregation allows the establishment of point-to-point links that have a higher aggregate bandwidth than the individual links that form the aggregation and the use of multiple systems at each end of the aggregation. This allows improved utilization of available links in Bridged local area network (LAN) environments, along with improved resilience in the face of failure of individual links or systems. In applications connecting separately administered networks, the networks are isolated from each other's fault recovery events.

## 1.3 State diagram conventions

This standard uses the state diagram conventions in Annex E of IEEE Std 802.1Q™-2018.[1]

When implementation of a state diagram is required for conformance, all states and transitions shown with solid lines shall be implemented. States and transitions shown with dashed lines may be implemented.

If a conflict exists between a state diagram and either the corresponding global transition tables or the textual description associated with the state machine, the state diagram takes precedence.



[1] Information on references can be found in Clause 2.

## 2. Normative references

The following referenced documents are indispensable for the application of this standard (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this standard is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 802®, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture.[2, 3]

IEEE Std 802.1AC™, IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Service Definition.

IEEE Std 802.1Q™, IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks.

IEEE Std 802.3™, IEEE Standard for Ethernet.

IETF RFC 1213 (IETF STD 17), Management Information Base for Network Management of TCP/IP-based internets: MIB-II, McCloghrie, K., and M. Rose, editors, Mar. 1991.[4]

IETF RFC 1321, The MD5 Message-Digest Algorithm, R. Rivest, Apr. 1992.

IETF RFC 2578 (STD 58), Structure of Management Information Version 2 (SMIv2), McCloghrie, K., D. Perkins, and J. Schoenwaelder, Apr. 1999.

IETF RFC 2579 (STD 58), Textual Conventions for SMIv2, McCloghrie, K., D. Perkins, J. Schoenwaelder, J. Case, M. Rose, and S. Waldbusser, Apr. 1999.

IETF RFC 2580 (STD 58), Conformance Statements for SMIv2, McCloghrie, K., D. Perkins, J. Schoenwaelder, J. Case, M. Rose, and S. Waldbusser, Apr. 1999.

IETF RFC 2863, The Interfaces Group MIB, McCloghrie, K., and F. Kastenholz, June 2000.

IETF RFC 3410, Introduction and Applicability Statements for Internet-Standard Management Framework, Case, J., R. Mundy, D. Partain, and B. Stewart, Dec. 2002.

IETF RFC 3414 (STD 62), User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3), Blumenthal, U., and B. Wijnen, Dec. 2002.

IETF RFC 3415 (STD 62), View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), Wijnen, B., R. Presuhn, and K. McCloghrie, Dec. 2002.

ISO/IEC 10165-4:1992, Information technology—Open Systems Interconnection—Structure of management information—Part 4: Guidelines for the definition of managed objects.[5]



[2] IEEE publications are available from The Institute of Electrical and Electronics Engineers (https://standards.ieee.org/).

[3] The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc.

[4] IETF documents (i.e., RFCs) are available from the Internet Engineering Task Force (https://www.rfc-archive.org/).

[5] ISO/IEC publications are available from the International Organization for Standardization (https://www.iso.org/), the International Electrotechnical Commission (https://www.iec.ch), and the American National Standards Institute (https://www.ansi.org/).

## 3. Definitions

For the purposes of this standard, the following terms and definitions apply. The *IEEE Standards Dictionary Online* can be consulted for terms not defined in this clause.[6]

This standard makes use of the following terms defined in IEEE Std 802:

— end station
— logical link
— station

This standard makes use of the following terms defined in IEEE Std 802.1AC:

— client (7.1 of IEEE Std 802.1AC-2016)
— frame (7.2 of IEEE Std 802.1AC-2016)
— Internal Sublayer Service (ISS) (Clause 11 of IEEE Std 802.1AC-2016)
— port (7.4 of IEEE Std 802.1AC-2016)

NOTE 1—It is helpful for the reader of this standard to review IEEE Std 802.1AC's description of the basic architectural concepts and terms used in this standard.[7]

This standard makes use of the following terms defined in IEEE Std 802.1Q:

— Bridge
— service instance

**Actor:** The local entity in a Link Aggregation Control Protocol exchange.

**Aggregateable:** A link terminating at an Aggregation Port that is configured to allow that link to form a Link Aggregation Group with other Aggregation Links.

**Aggregation Key:** A parameter, associated with each Aggregation Port and with each Aggregator of an Aggregation System, that prohibits the participation of multiple Aggregation Ports and Aggregators in the same Link Aggregation Group if their Aggregation Keys differ.

**Aggregation Link:** A local area network (LAN) that connects two Aggregation Systems.

**Aggregation Port:** A Service Access Point in an Aggregation System that provides access to a single Aggregation Link.

**Aggregation System:** A system that includes one or more Aggregators, each capable of using one or more Aggregation Ports to provide service to an Aggregator Client.

**Aggregator:** A multiplexing/demultiplexing entity that is capable of providing service at an Aggregator Port using the services provided by one or more Aggregation Ports.

**Aggregator Client:** A protocol entity that makes use of the service provided by a single Aggregator.

**Aggregator Port:** A port that provides access to the service provided by a single Aggregator.



[6] *IEEE Standards Dictionary Online* subscription is available at https://dictionary.ieee.org. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.
[7] Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

**conversation:** A set of frames transmitted from one end station to another, with the assumption that the communicating end stations require intermediate systems to maintain the ordering of those frames.

**Conversation ID:** An integer in the range of 0 through 4095, where each value identifies one or more conversations.

**Distributed Relay:** Transmission and reception of frames by Distributed Resilient Network Interconnect (DRNI) Gateway Ports in a pair of DRNI Systems using Aggregator Ports and Intra-Relay Ports in both DRNI Systems.

**Distributed Relay Client:** A protocol entity that makes use of the service provided by a DRNI Gateway.

**Distributed Resilient Network Interconnect (DRNI):** The DRNI Gateways in a pair of Aggregation Systems that are connected by an Intra-Relay Connection and can be used to create a single Link Aggregation Group using Aggregation Ports from either or both of the Aggregation Systems.

**Down frame:** A frame that has entered a Distributed Relay from the Distributed Relay Client.

**DRNI Gateway:** A protocol entity in an Distributed Resilient Network Interconnect (DRNI) System that can be paired with a DRNI Gateway in a another DRNI System and that provides Distributed Relay service at a DRNI Gateway Port using the services provided by an Aggregator Port in each system.

**DRNI Gateway Port:** A port that provides access to the service provided by a single Distributed Resilient Network Interconnect (DRNI) Gateway.

**DRNI System:** An Aggregation System containing one or more Distributed Resilient Network Interconnect (DRNI) Gateways.

**Gateway Conversation ID:** A Conversation identifier (ID) associated with a frame for the purpose of selecting a Distributed Resilient Network Interconnect (DRNI) Gateway Port.

**Home**: The local entity in a Distributed Relay Control Protocol (DRCP) exchange.

**Intra-Relay Connection (IRC):** A logical link that connects a pair of Distributed Resilient Network Interconnect (DRNI) Gateways comprising a Distributed Relay.

**Intra-Relay Port (IRP):** The service interface of a Distributed Resilient Network Interconnect (DRNI) Gateway provided by an Intra-Relay Connection.

**Key:** *See:* **Aggregation Key.**

**Link Aggregation Group (LAG):** A group of links that appear to an Aggregator Client as a single link.

**Link Number:** An Aggregation Link identifier that is unique within a Link Aggregation Group, with an operational value common to both the Actor and Partner Systems.

**Neighbor:** The remote entity in a Distributed Relay Control Protocol (DRCP) exchange.

**Partner:** The remote entity in a Link Aggregation Control Protocol (LACP) exchange.

**Port Conversation ID:** A Conversation identifier (ID) associated with a frame for the purpose of selecting an Aggregation Port.

**Service Identifier (Service ID):** A value extracted from the header of a frame (e.g., VID, I-SID) that identifies the service instance with which that frame is associated.

NOTE 2—This standard makes use of service identifiers specified in IEEE Std 802.1Q.

**Solitary:** An Aggregation Link that cannot be included in a Link Aggregation Group with any other Aggregation Link, due to administrative configuration at either of the attached Aggregation Systems.

**System:** When used in this standard and not preceded by “Aggregation” or “DRNI”, “System” means an Aggregation System.

**Up frame:** A frame that has entered a Distributed Relay through an Aggregator Port.

## 4. Acronyms and abbreviations

This standard contains the following acronyms and abbreviations:

| | |
|---|---|
| CID | Company ID[8] (see also OUI) |
| CSCD | Conversation-Sensitive Collection and Distribution |
| DA | destination address |
| DRCP | Distributed Relay Control Protocol |
| DRCPDU | Distributed Relay Control Protocol Data Unit |
| DRNI | Distributed Resilient Network Interconnect |
| DWC | Discard Wrong Conversation |
| FCS | frame check sequence |
| IRC | Intra-Relay Connection |
| IRP | Intra-Relay Port |
| ISS | Internal Sublayer Service |
| LACP | Link Aggregation Control Protocol |
| LACPDU | Link Aggregation Control Protocol Data Unit |
| LAG | Link Aggregation Group |
| LAG ID | Link Aggregation Group Identifier |
| LAN | local area network |
| LLC | logical link control |
| LLDP | Link Layer Discovery Protocol |
| MAC | medium access control |
| MIB | management information base |
| MSAP | MAC Service Access Point |
| NTT | Need To Transmit |
| OUI | organizationally unique identifier |
| PDU | Protocol Data Unit |
| PICS | protocol implementation conformance statement |
| SA | source address |
| TLV | type/length/value |
| TPMR | Two-Port MAC Relay |
| UCT | unconditional transition |



[8] See https://standards.ieee.org/develop/regauth/tut/eui.pdf.

# 5. Conformance

This clause specifies the mandatory and optional capabilities provided by conformant implementations of this standard.

## 5.1 Requirements terminology

For consistency with existing IEEE and IEEE 802.1™ standards, requirements placed upon conformant implementations of this standard are expressed using the following terminology:

a) ***Shall*** is used for mandatory requirements;
b) ***May*** is used to describe implementation or administrative choices ("may" means "is permitted to," and hence, "may" and "may not" mean precisely the same thing);
c) ***Should*** is used for recommended choices (the behaviors described by "should" and "should not" are both permissible but not equally desirable choices).

The protocol implementation conformance statement (PICS) proformas (see Annex A) reflect the occurrences of the words "shall," "may," and "should" within the standard.

The standard avoids needless repetition and apparent duplication of its formal requirements by using ***is***, ***is not***, ***are***, and ***are not*** for definitions and the logical consequences of conformant behavior. Behavior that is permitted but is neither always required nor directly controlled by an implementer or administrator, or whose conformance requirement is detailed elsewhere, is described by ***can***. Behavior that never occurs in a conformant implementation or System of conformant implementations is described by ***cannot***.

This clause lists requirements and options for certain capabilities, e.g., Link Aggregation. If a System claims to support such a capability for one of its ports, then the System is considered *conformant* if it meets that capability's requirements in this clause for that port. For example, a claim that a System supports Link Aggregation on a set of ports requires the System's behavior to meet the requirements specified in 5.3 for the identified ports.

## 5.2 Protocol implementation conformance statement

The supplier of an implementation that is claimed to conform to this standard shall provide the information necessary to identify both the supplier and the implementation, and shall complete a copy of the PICS proforma provided in Annex A.

## 5.3 Link Aggregation

### 5.3.1 Link Aggregation requirements

A conformant Link Aggregation System shall

a) Support the Link Aggregation Sublayer and conform to the state machines and procedures in 6.2.
b) Support the Link Aggregation Control Protocol (LACP) and conform to the state machines and procedures in 6.3 and 6.4.
c) Transmit and receive Link Aggregation Control Protocol Data Unit (LACPDUs) in the formats specified in 6.4.2.
d) Implement the Marker Responder as specified in 6.5.4.2 and respond to received Marker PDUs as specified in 6.5.1.
e) Support the LACP configuration for dual-homed Systems described in 6.7.5.

### 5.3.2 Link Aggregation options

A conformant Link Aggregation System may

a) Support the Marker Protocol and conform to the state machines and procedures in 6.5 and transmit and receive required Marker and Marker Response PDUs in the formats specified in 6.5.3.
b) Support the management functionality for Link Aggregation as specified in Clause 7.
c) Support SMIv2 management information base (MIB) modules for the management of Link Aggregation capabilities (Annex D).
d) Support Aggregation Port Debug Information package support as specified in 7.2.5.
e) Conform to the required specifications of the Link Layer Discovery Protocol (LLDP) of IEEE Std 802.1AB™ [B1].[9] A conformant System that supports LLDP shall provide an LLDP Parser/Multiplexer to attach zero or more instances of LLDP to each Aggregation Port as specified in F.1.
f) Implement a Protocol Parser/Multiplexer (6.1.3) for a protocol not explicitly specified in this standard.
g) Implement the techniques specified in 6.7.1 and 6.7.2 to form Link Aggregation Groups on an Aggregation System with the constraints described.
h) Support Conversation-Sensitive Collection and Distribution as specified in 6.2.7 and 6.6. An implementation that supports Conversation-Sensitive Collection and Distribution shall support a value of FORCE_FALSE for Admin_Discard_Wrong_Conversation and may support values of FORCE_TRUE and AUTO.
i) Support Per-Service Frame Distribution as specified in 8.2 and shall
    1) Support Conversation-Sensitive Collection and Distribution as specified in 6.2.7 and 6.6, and
    2) Transmit and receive version 2 LACPDUs as specified in 6.4.2.4, and
    3) Support classification by Customer VLAN Tag for C-tagged interfaces and classification by Service VLAN tag for S-tagged and B-tagged interfaces,

    and may
    4) Support classification by Backbone Service Instance Tag as specified in 8.2.3.

## 5.4 Distributed Resilient Network Interconnect (DRNI)

### 5.4.1 DRNI requirements

A conformant DRNI System shall

a) Conform to the requirements specified for Link Aggregation as specified in 5.3.
b) Support a DRNI consisting of two DRNI Systems as specified in Clause 9 and shall
    1) Support a DRNI Gateway having a single Intra-Relay Port (IRP) and conform to the state machines and procedures in 9.5,
    2) Support the Distributed Relay Control Protocol (DRCP) and conform to the state machines and procedures specified in 9.6, and
    3) Transmit and receive required Distributed Relay Control Protocol Data Units (DRCPDUs) in the formats specified in 9.6.2.
c) Support using a dedicated link for the Intra-Relay Connection (IRC) as specified in 9.3.

[9] The numbers in brackets correspond to the numbers in the bibliography in Annex G.

### 5.4.2 DRNI options

A conformant DRNI System may

a) Implement any of the options specified for Link Aggregation (5.3.2).
b) Support using a LAG for the IRC by supporting the IRP with a dedicated Aggregator as specified in 9.3.
c) Support using a method other than a dedicated link or LAG for the IRC as specified in 9.3.
d) Support Organization-Specific TLVs in DRCPDUs as specified in 9.6.2.7.

# 6. Link Aggregation

## 6.1 Overview

This clause defines an optional Link Aggregation sublayer for use with links offering the IEEE 802.1AC Internal Sublayer Service (ISS). Link Aggregation allows one or more point-to-point links to be aggregated together to form a Link Aggregation Group (LAG) so that an Aggregator Client can treat the LAG as if it were a single link.

The models of operation in this clause provide a basis for specifying the externally observable behavior of the operation and are not intended to place additional constraints on implementations; these can adopt any internal model of operation compatible with the externally observable behavior specified. Conformance of equipment to this standard is purely with respect to observable protocol.

### 6.1.1 Goals and objectives

Link Aggregation, as specified in this clause, provides the following:

a) **Increased bandwidth**—The capacity of multiple links is combined into one logical link.
b) **Increased availability**—The failure or replacement of a single link within a LAG need not cause failure from the perspective of an Aggregator Client.
c) **Load sharing**—Aggregator Client traffic can be distributed across multiple links.
d) **Automatic configuration**—In the absence of manual overrides, an appropriate set of LAGs is automatically configured, and individual links are allocated to those groups. If a set of links can aggregate, they will aggregate.
e) **Rapid configuration and reconfiguration**—In the event of changes in physical connectivity, Link Aggregation will quickly converge to a new configuration, typically on the order of milliseconds for link down events and 1 s or less for link up events.

NOTE 1—Timing out LACPDU exchanges is the method of last resort for detecting link failures and shifting data flows to other physical links. The MAC_Operational status generated by link hardware is the first choice for link failure detection. Connectivity Fault Management (CFM) (Clause 18 of IEEE Std 802.1Q-2018) provides a method for detecting link failures (also indicated via the MAC_Operational status) when hardware means are not applicable.

f) **Deterministic behavior**—The link aggregation configuration can be made independent of past events and completely determined by the connectivity provided by the individual links and the administrative configuration of the connected systems.
g) **Low risk of duplication or misordering**—During both steady-state operation and link (re)configuration, there is a low probability that frames are duplicated or misordered.
h) **Support of existing medium access control (MAC) Clients**—No change is required to existing higher-layer protocols or applications to use Link Aggregation.
i) **Backwards compatibility with aggregation-unaware devices**—Links that cannot take part in Link Aggregation because of their inherent capabilities, management configuration, or the capabilities of the devices to which they attach operate as normal, individual links.
j) **Accommodation of differing capabilities and constraints**—Devices with differing hardware and software constraints on Link Aggregation are, to the extent possible, accommodated.
k) **No change to frame formats**—Link Aggregation neither adds to, nor changes the contents of, frames exchanged between Aggregator Clients.
l) **Network management support**—The standard specifies appropriate management objects for configuration, monitoring, and control of Link Aggregation.
m) **Point-to-point aggregations**—The mechanisms specified in this clause form a Link Aggregation Group providing a point-to-point connectivity service.

n) **Dissimilar MACs**—Link Aggregation can be used over any physical or logical medium supporting the ISS with a service providing point-to-point connectivity (i.e., the ISS operPointToPoint variable is TRUE).

Aggregating different data rates is not prohibited nor required by this standard. Determining how to distribute traffic across links of different data rates is beyond the scope of this standard.

NOTE 2—Previous versions of this standard restricted the data rate of the aggregated links. However, the mechanisms specified did not depend on whether the links operated at the same rate.

### 6.1.2 Positioning of Link Aggregation within the IEEE 802 architecture

Link Aggregation comprises an optional sublayer that (at its upper boundary) has an ISS to Aggregator Clients attached to Aggregator Ports and (at its lower boundary) has an ISS for each of its Aggregation Ports. Figure 6-1 depicts a common case where Aggregation Ports in a system are supported directly by a MAC.

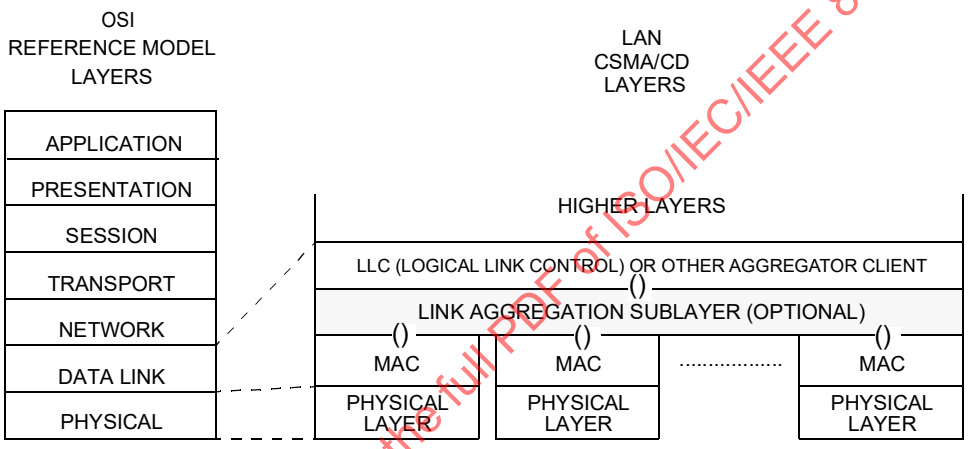



**Figure 6-1—Architectural positioning of Link Aggregation sublayer**

It is possible to implement the optional Link Aggregation sublayer for some ports within a System while not implementing it for other ports; i.e., it is not necessary for all ports in a System to be subject to Link Aggregation. A conformant implementation is not required to be able to apply the Link Aggregation sublayer to every port.

NOTE—IEEE Std 802.1AX-2008 and Clause 43 in earlier editions of IEEE Std 802.3 used the term "MAC Client" instead of the term "Aggregator Client" and placed Link Aggregation between the IEEE 802.3 MAC sublayer and the IEEE 802.3 MAC Control sublayer. IEEE Std 802.1AX-2014 repositioned Link Aggregation as a sublayer above the MAC sublayer. This rearrangement aligned the standard with common industry practice. It did not cause implementations conformant to earlier editions of the standard to become non-interoperable with those conformant to IEEE Std 802.1AX-2014 and subsequent editions.

### 6.1.3 Protocol Parser/Multiplexer

A Protocol Parser/Multiplexer is a functional element that separates specific control protocol frames from a sequence of M_UNITDATA.indications received at an ISS interface and combines frames generated for that control protocol with a sequence of M_UNITDATA.requests at an ISS interface. The general purpose Protocol Parser/Multiplexer described here can be customized to a specific control protocol, of which there are three examples in this standard:

a) The LLDP Parser/Multiplexer (F.1.1)
b) The LACP Parser/Multiplexer (6.2.9)
c) The DRCP Parser/Multiplexer (9.5.1)

NOTE—The Aggregator Parser/Multiplexer (6.2.7) and DR_Aggregator Parser/Multiplexer (9.5.2.6) are not instances of a Protocol Parser/Multiplexer. These perform additional parsing functions not included in the general purpose Protocol Parser/Multiplexer and are described by their own state diagrams.

A Protocol Parser/Multiplexer has three service interfaces, each an instance of the ISS, as follows:

d) One DownPort makes use of an instance of the ISS to pass data and control frames to and from lower layers in the interface stack.
e) One DataPort offers an instance of the ISS to higher layers in the interface stack.
f) One ControlPort offers an instance of the ISS to the functional element for the control protocol.

A specific instance of the Protocol Parser/Multiplexer, e.g., the LACP Parser/Multiplexer (6.2.9), ties these three notional service interfaces to specific service interfaces, e.g., those in 6.2.2, and provides a definition of the IsControlFrame function (6.1.3.1.2).

On transmission, the Protocol Multiplexer provides transparent pass-through of frames from the ControlPort or the DataPort to the DownPort.

On receipt of a frame from its DownPort, the Protocol Parser uses the IsControlFrame function to determine whether the frame is a control frame. It passes control frames up through the ControlPort and passes all other frames up through the DataPort. The Protocol Parser implements the function specified by the state diagram shown in Figure 6-2 and the associated definitions contained in 6.1.3.1.

### 6.1.3.1 Protocol Parser state diagram

#### 6.1.3.1.1 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

msdu_type
: The type value of the mac_service_data_unit that is either an EtherType or an LLC Address, determined as described in Clause 9 of IEEE Std 802-2014.

BEGIN
: A Boolean variable that is set to TRUE when the System is initialized or reinitialized and is set to FALSE when (re)initialization has completed.
Data Type: Boolean

#### 6.1.3.1.2 Functions

IsControlFrame
: Given an input frame (DA, SA, mac_service_data_unit, priority), returns TRUE if the frame is a control frame for the ControlPort or FALSE if it is a data frame for the DataPort.
Value returned: Boolean

#### 6.1.3.1.3 Messages

ControlPort:M_UNITDATA.indication
DataPort:M_UNITDATA.indication
DownPort:M_UNITDATA.indication

The service primitives used to pass a received frame to a client with the specified parameters.

#### 6.1.3.1.4 State diagram

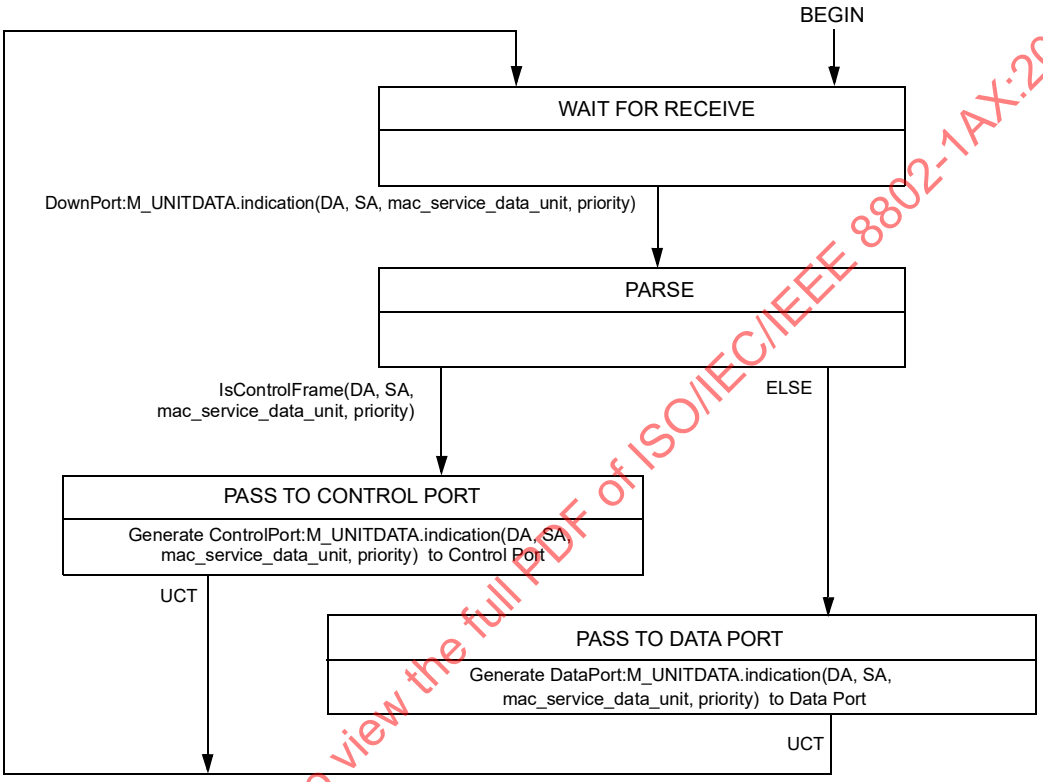



**Figure 6-2—Protocol Parser state diagram**



## 6.2 Link Aggregation operation

Figure 6-3 depicts the major blocks that form the Link Aggregation sublayer and their interrelationships. The Link Aggregation sublayer comprises the following functions:

a) **Frame Distribution.** This block is responsible for taking frames submitted by the Aggregator Client and submitting them for transmission on the appropriate Aggregation Port, based on a frame distribution algorithm employed by the Frame Distributor. Frame Distribution includes a Frame Distributor and an optional Marker Generator/Receiver used for the Marker protocol. (See 6.2.4, 6.2.5, and 6.5.)

b) **Frame Collection**. This block is responsible for passing frames received from the various Aggregation Ports to the Aggregator Client. Frame Collection includes a Frame Collector and a Marker Responder, used for the Marker protocol. (See 6.2.3, 6.2.6, and 6.5.)

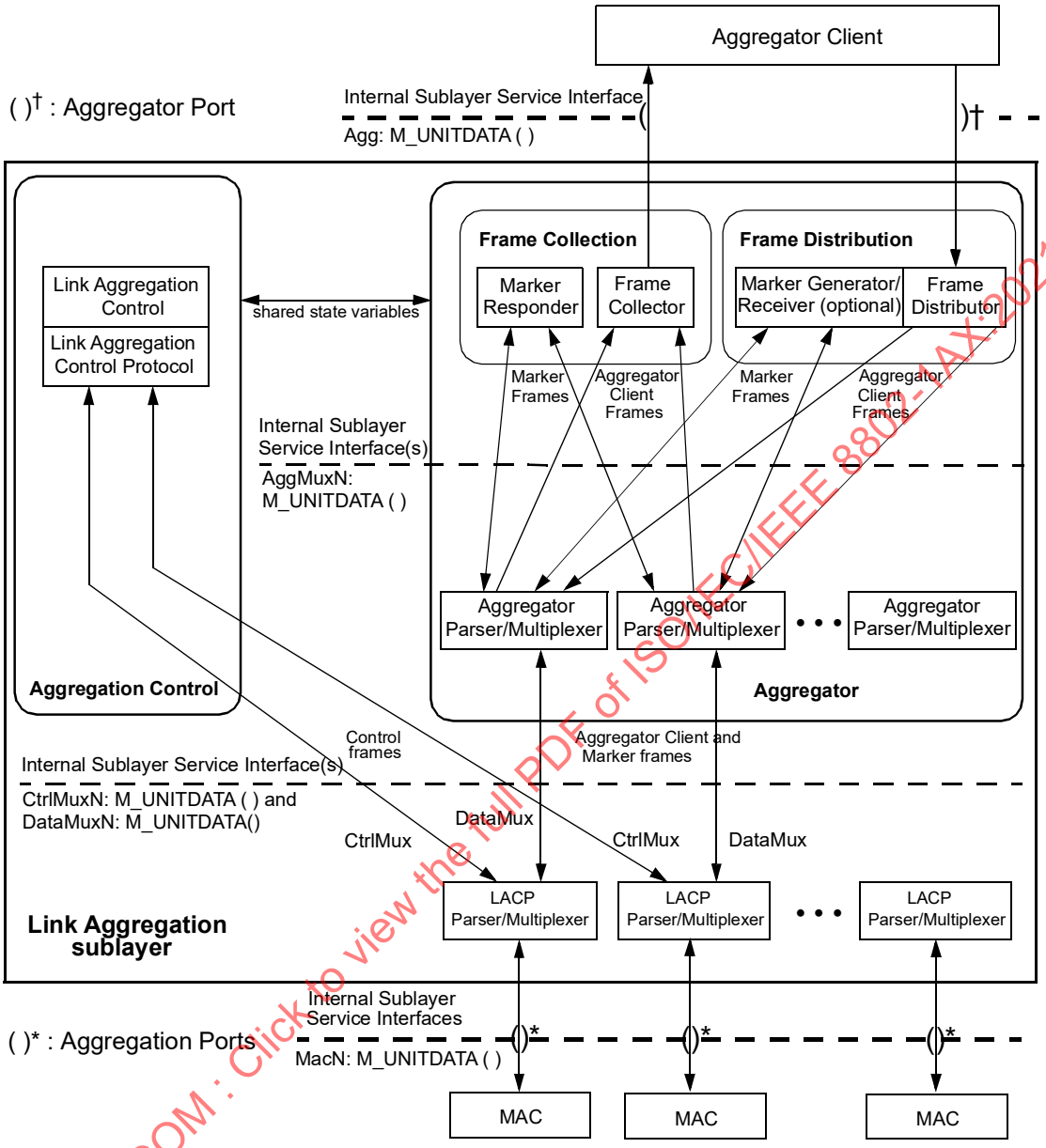



**Figure 6-3—Link Aggregation sublayer block diagram**

c) **Aggregator Parser/Multiplexers.** On transmit, these entities simply pass frame transmission requests from the Frame Distributor, Marker Generator, and/or Marker Responder to the appropriate Aggregation Port. On receive, these entities distinguish among Marker Request, Marker Response, and MAC Protocol Data Units and pass each to the appropriate entity (Marker Responder, Marker Receiver, and Frame Collector, respectively).

d) **Aggregator**. The combination of Frame Distribution and Frame Collection, along with the Aggregator Parser/Multiplexers, is referred to as the Aggregator.

e) **Aggregation Control**. This block is responsible for the configuration and control of Link Aggregation. It incorporates a Link Aggregation Control Protocol (LACP) that can be used for automatic communication of aggregation capabilities between Systems and automatic configuration of Link Aggregation.

f) **LACP Parser/Multiplexers**. On transmit, these entities simply pass frame transmission requests from the Aggregator and Control entities to the appropriate Aggregation Port. On receipt, these entities distinguish Link Aggregation Control PDUs from other frames, passing the LACPDUs to the appropriate sublayer entity, and all other frames to the Aggregator.

### 6.2.1 Principles of Link Aggregation

Link Aggregation allows an Aggregator Client to treat a set of one or more Aggregation Ports as if it were a single port. In doing so, it employs the following principles and concepts:

a) An Aggregator Client communicates with a set of Aggregation Ports through an Aggregator, which presents a standard ISS interface to the Aggregator Client. One or more Aggregation Ports attach to an Aggregator within a System.

b) It is the responsibility of the Aggregator to distribute frame transmissions from the Aggregator Client to the various Aggregation Ports, and to collect received frames from the Aggregation Ports and pass them to the Aggregator Client transparently.

c) A System can contain multiple Aggregators, serving multiple Aggregator Clients. A given Aggregation Port will attach to (at most) a single Aggregator at any time. An Aggregator Client is served by a single Aggregator at a time.

d) The attachment of Aggregation Ports to Aggregators within a System is managed by the Link Aggregation Control function for that System, which is responsible for determining which links can be aggregated, aggregating them, attaching the Aggregation Ports within the System to an appropriate Aggregator, and monitoring conditions to determine when a change in aggregation is needed.

e) Such determination and attachment might be under manual control through direct manipulation of the state variables of Link Aggregation (e.g., Keys) by a network manager. In addition, automatic determination, configuration, attachment, and monitoring can occur through the use of a Link Aggregation Control Protocol (LACP). LACP uses peer exchanges across the links to determine, on an ongoing basis, the aggregation capability of the various links and continuously provides the maximum level of aggregation capability achievable between a given pair of Systems.

f) Frame ordering has to be maintained for certain sequences of frame exchanges between Aggregator Clients (known as conversations, see Clause 3). The Frame Distributor ensures that all frames of a given conversation are passed to a single Aggregation Port. For any given Aggregation Port, the Frame Collector is required to pass frames to the Aggregator Client in the order that they are received from that Aggregation Port. The Frame Collector is otherwise free to select frames received from the Aggregation Ports in any order. Since there are no means for frames to be misordered on a single link, this guarantees that frame ordering is maintained for any conversation.

g) Conversations can be moved among Aggregation Ports within an aggregation, both for load balancing and to maintain availability in the event of link failures. Frame ordering can be maintained (6.5 and 6.6) when conversations are moved.

h) This standard does not require the use of a particular distribution algorithm.

i) Each Aggregation Port is assigned a MAC address, unique over the LAG and the IEEE 802.1Q Bridged LAN (if any) to which the LAG is connected. This MAC address is used as the source address (SA) for frame exchanges that are initiated by entities within the Link Aggregation sublayer itself (i.e., LACP and Marker protocol exchanges).

NOTE—LACP and Marker Protocol use a multicast DA for all exchanges and do not require an Aggregation Port to recognize more than one unicast address on received frames.

j) Each Aggregator is assigned a MAC address, unique over the LAG and the IEEE 802.1Q Bridged LAN (if any) to which the LAG is connected; this address is used as the MAC address of the aggregation from the perspective of the Aggregator Client, both as a SA for transmitted frames and as the destination address (DA) for received frames. The MAC address of the Aggregator can be one of the MAC addresses of an Aggregation Port in the associated LAG (see 6.2.10).

### 6.2.2 Service interfaces

The Aggregator Client communicates with the Aggregator using the ISS specified in IEEE Std 802.1AC. Similarly, Link Aggregation communicates internally (between Frame Collection/Distribution, the Aggregator Parser/Multiplexers, the LACP Parser/Multiplexers, and Link Aggregation Control) and with its attached Aggregation Ports using the same service interface. No new interlayer service interfaces are defined for Link Aggregation.

Since Link Aggregation uses several instances of the ISS, it is necessary to introduce a notation convention so that the reader can be clear about which interface is being referenced at any given time. A prefix is therefore assigned to each service primitive to indicate which of the interfaces is being invoked, as depicted in Figure 6-3. The prefixes are as follows:

a) Agg:, for primitives issued on the interface between the Aggregator Client and the Link Aggregation sublayer.
b) AggMuxN:, for primitives issued on the interface between Aggregator Parser/Multiplexer N and its internal clients (where N is the Port Number associated with the Aggregator Parser/Multiplexer).
c) CtrlMuxN:, for primitives issued on the interface between LACP Parser/Multiplexer N and Link Aggregation Control (where N is the Port Number associated with the LACP Parser/Multiplexer).
d) DataMuxN:, for primitives issued on the interface between LACP Parser/Multiplexer N and Aggregator Parser/Multiplexer N (where N is the Port Number associated with the LACP Parser/Multiplexer).
e) MacN:, for primitives issued on the interface between underlying MAC N and its LACP Parser/Multiplexer (where N is the Port Number associated with the underlying MAC).

Aggregator Clients can generate Agg:M_UNITDATA.request primitives for transmission on an aggregated link. These are passed by the Frame Distributor to an Aggregation Port selected by the distribution algorithm. MacN:M_UNITDATA.indication primitives signifying received frames are passed unchanged from an Aggregation Port to the Aggregator Client by the Frame Collector.

In order to satisfy the needs of higher layers to access physical ports directly, the Aggregator Parser/Multiplexer (6.2.7) is provided. In particular, if IEEE Std 802.1AB [B1] is implemented in a System supporting Link Aggregation, the Link Aggregation operation (6.2) can attach zero or more instances of LLDP to each Aggregation Port.

### 6.2.3 Frame Collector

A Frame Collector is responsible for receiving incoming frames (i.e., AggMuxN:M_UNITDATA.indications) from the set of individual links that form the LAG (through each link's associated Aggregator Parser/Multiplexer) and delivering them to the Aggregator Client. Frames received from a given Aggregation Port are delivered to the Aggregator Client in the order that they are received by the Frame Collector. Since the Frame Distributor is responsible for maintaining any frame ordering constraints, there is no requirement for the Frame Collector to perform any reordering of frames received from multiple links.

The Frame Collector shall implement the function specified in the state diagram shown in Figure 6-4 and the associated definitions contained in 6.2.3.1.

#### 6.2.3.1 Frame Collector state diagram

##### 6.2.3.1.1 Constants

CollectorMaxDelay
: In tens of microseconds, the maximum time that the Frame Collector can delay the delivery of a frame received from an Aggregator Parser to its Aggregator Client. Value is assigned by management or administration policy.
Data Type: Integer
Alias of aAggCollectorMaxDelay (7.3.1.1.32)

##### 6.2.3.1.2 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

BEGIN
: A Boolean variable that is set to TRUE when the System is initialized or reinitialized and is set to FALSE when (re)initialization has completed.
Data Type: Boolean

##### 6.2.3.1.3 Messages

Agg:M_UNITDATA.indication
AggMuxN:M_UNITDATA.indication
: The service primitives used to pass a received frame to a client with the specified parameters.

##### 6.2.3.1.4 State diagram

The architectural models of ISS and Link Aggregation do not make any provision for queuing of frames between the link and the Aggregator Client. However, practical implementations of Link Aggregation will typically incur both queuing and delay in the Frame Collector. In order to ensure that frame delivery is not delayed indefinitely (which could cause a frame ordering problem when moving conversations from one link to another), the Frame Collector shall, upon receiving a frame from an Aggregator Parser, either deliver the frame to its Aggregator Client, or discard the frame within a CollectorMaxDelay time. The Frame Distributor (within the Partner System at the other end of the link) can assume that all frames transmitted on a given link have been either received by its Partner's Aggregator Client or discarded after a CollectorMaxDelay plus the propagation delay of the link. The use of CollectorMaxDelay is further discussed in B.3.

NOTE—Because frame discard due to CollectorMaxDelay is a function of queuing and other entities not specified in this standard, the discard action is a requirement of the system, not just of the Frame Collector, and is therefore not shown in Figure 6-4.

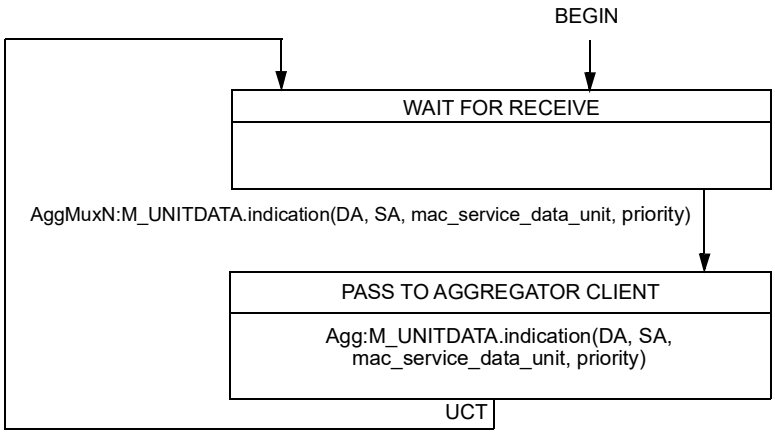



**Figure 6-4—Frame Collector state diagram**

### 6.2.4 Frame Distributor

The Frame Distributor is responsible for taking outgoing frames from the Aggregator Client and transmitting them through the set of links that form the LAG. The Frame Distributor implements a distribution function (algorithm) responsible for choosing the link to be used for the transmission of any given frame or set of frames.

This standard allows a wide variety of distribution algorithms. However, practical frame distribution algorithms do not misorder frames that are part of any given conversation, nor do they duplicate frames.

Rather, frame order is maintained by ensuring that all frames that compose a given conversation are transmitted on a single link in the order that they are generated by the Aggregator Client. No addition (or modification) of any information to the MAC frame is necessary, nor is any buffering or processing on the part of the corresponding Frame Collector in order to reorder frames. This approach permits a wide variety of distribution and load balancing algorithms to be used, while also ensuring interoperability between devices that adopt differing algorithms.

NOTE—Distribution algorithms and maintenance of frame ordering are discussed in Annex B.

Conversation-Sensitive Collection and Distribution (6.6) provides a structure for identifying various distribution algorithms, configuring the distribution algorithm to be used, and conveying that information to the LACP partner in LACP Version 2 TLVs (6.4.2.4). Per-Service Frame Distribution (Clause 8) specifies a set of distribution algorithms that can be used by a Link Aggregation implementation.

The Frame Distributor shall implement the function specified in the state diagram shown in Figure 6-5 and the associated definitions contained in 6.2.4.1.

#### 6.2.4.1 Frame Distributor state diagram

##### 6.2.4.1.1 Variables

DA
SA
mac_service_data_unit
priority
The parameters of the M_UNITDATA.request primitive.

BEGIN
A Boolean variable that is set to TRUE when the System is initialized or reinitialized and is set to FALSE when (re)initialization has completed.
Data Type: Boolean

#### 6.2.4.1.2 Messages

Agg:M_UNITDATA.request
AggMuxN:M_UNITDATA.request
The service primitives used to transmit a frame with the specified parameters at the Aggregation Port, N, determined by the distribution algorithm. When Conversation-Sensitive Collection and Distribution is not supported, the algorithm used to determine the value of N is unspecified. When Conversation-Sensitive Collection and Distribution is supported, the algorithm used to determine the value of N is specified by the Actor_Port_Algorithm variable. It is possible that the value of N will be zero, indicating that the service primitive is to be discarded.

#### 6.2.4.1.3 State diagram

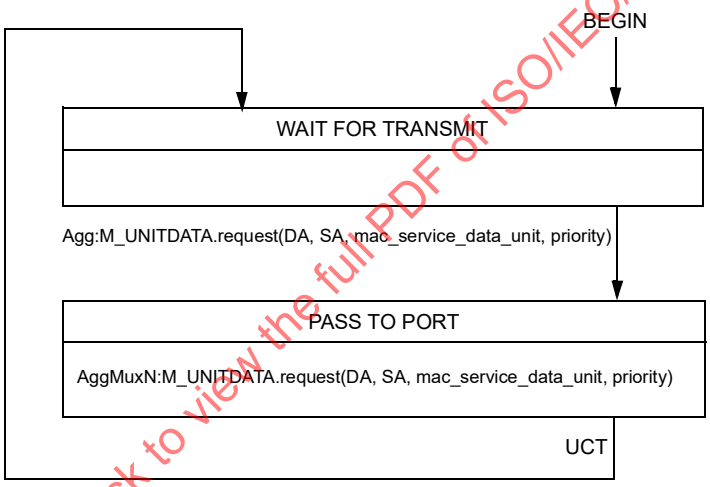



**Figure 6-5—Frame Distributor state diagram**

### 6.2.5 Marker Generator/Receiver (optional)

The optional Marker Generator is used by the Marker protocol, as specified in 6.5. When implemented and so requested by the distribution algorithm, the Marker Generator shall issue an AggMuxN:M_UNITDATA.request primitive, with a mac_service_data_unit containing a Marker PDU as defined in 6.5.3, to the Aggregation Port associated with the conversation being marked, subject to the timing restrictions for Slow Protocols specified in IEEE Std 802.3-2018, Annex 57A.

The optional Marker Receiver is used by the Marker protocol, as specified in 6.5. It receives Marker Response PDUs from the Aggregator Parser.

### 6.2.6 Marker Responder

The Marker Responder is used by the Marker protocol, as specified in 6.5. The Marker Responder receives Marker PDUs (generated by a Partner System's Marker Generator), and transmits a Marker Response PDU through the same Aggregation Port from which the Marker PDU was received. While implementation of the Marker Generator/Receiver is optional, the ability to respond to a Marker PDU (the Marker Responder) is mandatory. An implementation conformant to this clause that is not under a Distributed Relay component (DRNI Gateway, see 9.2, 9.5) shall implement the Marker Responder as specified in 6.5.4.2, thus ensuring that implementations that need to make use of the protocol can do so.

A Marker Responder that is under a DRNI Gateway is not required to respond to a Marker PDU.

NOTE—Since this standard lacks any means for coordinating Marker PDU responses among the DRNI Systems comprising a DRNI, it is safer to ignore a Marker PDU and allow the sender to time out, rather than to return a Marker PDU and run the risk of delivering frames out of order.

### 6.2.7 Aggregator Parser/Multiplexer

On transmission, the Aggregator Multiplexer shall provide transparent pass-through of frames submitted by the Marker Responder and optional Marker Generator to the Aggregation Port specified in the transmission request. The Aggregator Multiplexer shall provide transparent pass-through of frames submitted by the Frame Distributor to the Aggregation Port specified in the transmission request only when the Aggregation Port state is Distributing (see 6.4.13). Furthermore, if Conversation-Sensitive Collection and Distribution is supported, the value of the Distribution_Conversation_Mask for the Port_Conversation_ID associated with the frame is TRUE. Otherwise, such frames shall be discarded.

On receipt, the Aggregator Parser decodes frames received from the LACP Parser, passes those frames destined for the Marker Responder or Marker Receiver to the selected entity, and discards frames with invalid Slow Protocol subtype values (see IEEE Std 802.3-2018, Table 57A–2). The Aggregator Parser shall pass all other frames to the Frame Collector for passage to the Aggregator Client only when the Aggregation Port state is Collecting (see 6.4.13). Furthermore, if Conversation-Sensitive Collection and Distribution is supported, the value of the Collection_Conversation_Mask for the Port_Conversation_ID associated with the frame is TRUE. Otherwise, such frames shall be discarded. The Aggregator Parser shall implement the function specified in the state diagram shown in Figure 6-6 and the associated definitions contained in 6.2.7.1.

#### 6.2.7.1 Aggregator Parser/Multiplexer state diagrams

##### 6.2.7.1.1 Constants

Slow_Protocols_Type
: The value of the Slow Protocols EtherType (Table 6-3).

Marker_subtype
: The value of the Subtype field for the Marker protocol. (See 6.5.3.)
: Data Type: Integer
: Value: 2

Marker_Information
: The encoding of the Marker Information TLV_Type field. (See 6.5.3.)
: Data Type: Integer
: Value: 1

Marker_Response_Information
: The encoding of the Marker Response Information TLV_Type field. (See 6.5.3.)
: Data Type: Integer
: Value: 2

#### 6.2.7.1.2 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

Protocol_DA
: The address determined by the setting of the aAggPortProtocolDA managed object (7.3.2.2.1). A particular instance of link aggregation shall use the same DA for LACP as it does for the Marker protocol.
Data Type: 6 octets.

Length/Type
: The value of the Length/Type field in a received frame.
Data Type: Integer

Subtype
: The value of the octet following the Length/Type field in a Slow Protocol frame. (See IEEE Std 802.3-2018, Annex 57A.)
Data Type: Integer

TLV_Type
: The value contained in the octet following the Version Number in a received Marker or Marker Response frame. This identifies the “type” for the type/length/value (TLV) tuple. (See 6.5.3.)
Data Type: Integer

Collecting
: An alias of Actor_Oper_Port_State.Collecting. When TRUE, the Aggregator Parser can pass frames to the Collector.
Data Type: Boolean.

Collection_Conversation_Mask
: A Boolean vector, indexed by Port_Conversation_ID, that indicates whether the frame is allowed to reach the Aggregator when received through this Aggregation Port (TRUE = passes). When Conversation-Sensitive Collection and Distribution is not supported, this variable is not used (implicitly contains the value TRUE for all Port_Conversation_IDs). When Conversation-Sensitive Collection and Distribution is supported, the value is determined by the Update Mask machine (6.6.3.5).
Data Type: sequence of Boolean values, indexed by Port_Conversation_ID.
Alias of aAggPortOperConversationCollected (7.3.2.1.26).

Distributing
: An alias of Actor_Oper_Port_State.Distributing. When TRUE, the Aggregator Multiplexer can pass frames to the Aggregation Port.
Data Type: Boolean.

Distribution_Conversation_Mask
: A Boolean vector, indexed by Port_Conversation_ID, that indicates whether the M_UNITDATA.request is allowed to be passed to the Aggregation Port (TRUE = passes). When Conversation-Sensitive Collection and Distribution is not supported, this variable is not used (implicitly contains the value TRUE for all Port_Conversation_IDs). When Conversation-Sensitive Collection and Distribution is supported, the value is determined by the Update Mask machine (6.6.3.5).
Data Type: sequence of Boolean values, indexed by Port_Conversation_ID.
Alias of aAggPortOperConversationPasses (7.3.2.1.25).

Port_Conversation_ID
: A Conversation Identifier (6.6.1) derived from the parameters of the service primitive. When Conversation-Sensitive Collection and Distribution is supported, the mechanism to derive the Conversation Identifier is specified by the Actor_Port_Algorithm (6.6.3.1).
Data Type: Integer
Value: 0 to 4095

BEGIN
: A Boolean variable that is set to TRUE when the System is initialized or reinitialized and is set to FALSE when (re)initialization has completed.
Data Type: Boolean

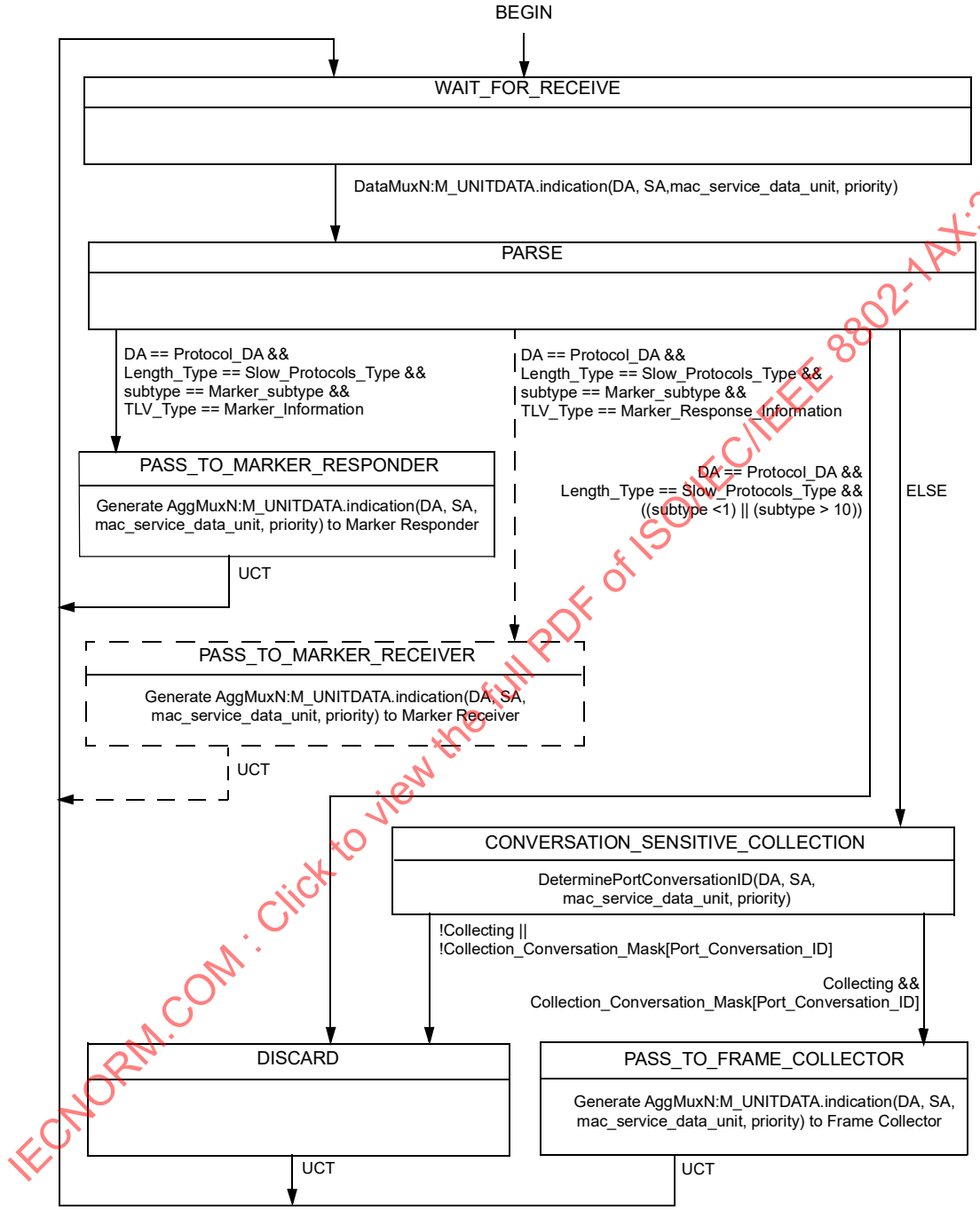
#### 6.2.7.1.3 Functions

DeterminePortConversationID
: When Conversation-Sensitive Collection and Distribution is supported, this function determines a Port_Conversation_ID for the frame as specified by the Actor_Port_Algorithm variable using the mechanisms described in 6.6 and Clause 8.

#### 6.2.7.1.4 Messages

DataMuxN:M_UNITDATA.indication
AggMuxN:M_UNITDATA.indication
: The service primitives used to pass a received frame to a client with the specified parameters.

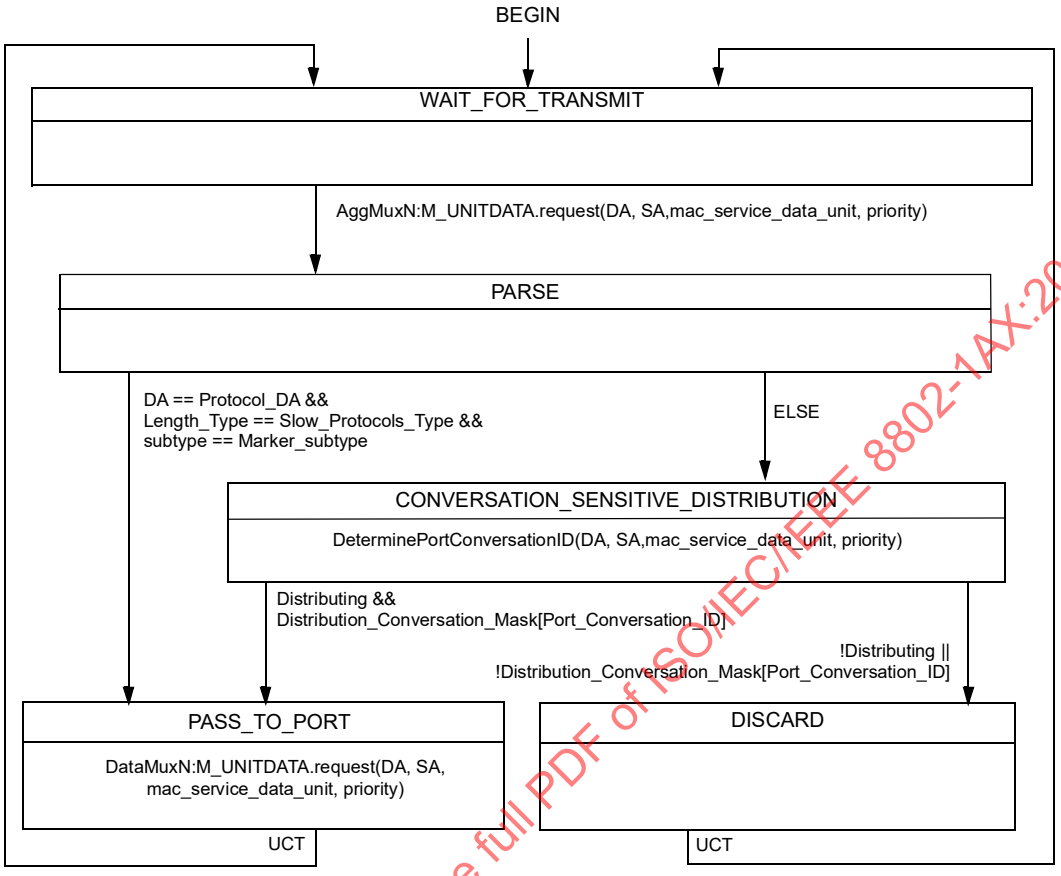
#### 6.2.7.1.5 State diagrams






If the optional Marker Receiver is not implemented, Marker Responses go from the PARSE state to the CONVERSATION_SENSITIVE_COLLECTION state.
If Conversation-Sensitive Collection and Distribution is not implemented, transitions from the CONVERSATION_SENSITIVE_COLLECTION state are based solely on the value of the Collecting variable.

**Figure 6-6—Aggregator Parser state diagram**




If Conversation-Sensitive Collection and Distribution is not implemented, transitions from the CONVERSATION_SENSITIVE_DISTRIBUTION state are based solely on the value of the Distributing variable.

**Figure 6-7—Aggregator Multiplexer state diagram**

### 6.2.8 Aggregator

An Aggregator comprises an instance of a Frame Collection function, an instance of a Frame Distribution function, and one or more instances of the Aggregator Parser/Multiplexer function for a LAG. A single Aggregator is associated with each LAG. An Aggregator offers an ISS interface to its associated Aggregator Client; access to the physical media by an Aggregator Client is always achieved via an Aggregator.

A single individual MAC address is associated with each Aggregator (see 6.2.10).

If the following are all true, an Aggregator is available for use by the Aggregator Client, and MAC_Operational at the Aggregator ISS is TRUE (6.3.12):

a) The Aggregator has one or more attached Aggregation Ports.
b) The Aggregator has not been set to a disabled state by administrative action (see 7.3.1.1.13).
c) One or more of the attached Aggregation Ports is Collecting and Distributing (see 7.3.1.1.14).

NOTE—To simplify the modeling and description of the operation of Link Aggregation, it is assumed that there are as many Aggregators as there are Aggregation Ports in a given System; however, this is not a requirement of this standard. Aggregation of two or more Aggregation Ports consists of changing the attachments between Aggregation Ports and Aggregators so that more than one Aggregation Port is attached to a single Aggregator. The creation of any

aggregations of two or more links therefore results in one or more Aggregators that are attached to more than one Aggregation Port and one or more Aggregators that are not attached to any Aggregation Port. An Aggregator that is not attached to any Aggregation Port appears to an Aggregator Client as a MAC interface to an inactive Aggregation Port. During times when the attachments between Aggregation Ports and Aggregators are changing, or as a consequence of particular configuration choices, there might be occasions when one or more Aggregation Ports are not attached to any Aggregator.

### 6.2.9 LACP Parser/Multiplexer

There is an LACP Parser/Multiplexer for each Aggregation Port. Each LACP Parser/Multiplexer is an instance of the Protocol Parser/Multiplexer described in 6.1.3. Specifically:

a) The DownPort of the LACP Parser/Multiplexer is MacN (6.2.2) for LACP Parser/Multiplexer N.
b) The ControlPort of the LACP Parser/Multiplexer is CtrlMuxN (6.2.2) for LACP Parser/Multiplexer N.
c) The DataPort of the LACP Parser/Multiplexer is DataMuxN (6.2.2) for LACP Parser/Multiplexer N.
d) The IsControlFrame function returns a value of TRUE if the DA of the service primitive is equal to the aAggPortProtocolDA (7.3.2.2.1) value, the msdu_type is the Slow_Protocols EtherType (Table 6-3), the subtype is the LACP_subtype (IEEE Std 802.3-2018, Table 57A–3), and the length of the mac_service_data_unit is greater than or equal to 60 octets.

### 6.2.10 Addressing

Each IEEE 802 MAC has an associated individual MAC address regardless of whether that MAC is used for Link Aggregation.

Each Aggregator to which one or more Aggregation Ports are attached has an associated individual MAC address (see 6.3.3). The MAC address of the Aggregator can be the individual MAC address of one of the MACs in the associated LAG, or it can be a distinct MAC address. The manner in which such addresses are chosen is not otherwise constrained by this standard.

#### 6.2.10.1 Source address (SA)

The source MAC address of LACPDUs and Marker protocol frames originating within the Link Aggregation sublayer is that of the Aggregation Port used for their transmission. The source MAC address of frames transmitted by the Aggregator Client is that of the Aggregator (for frames originating from the Aggregator Client) or that assigned by the originator of the frame (for frames bridged by the Aggregator Client).

NOTE—This behavior causes both the Aggregation Port and the Aggregator to behave the same way as a standard MAC with regard to frames submitted by their clients.

#### 6.2.10.2 Destination address (DA)

The destination MAC address of LACPDUs and Marker protocol frames originating within the Link Aggregation sublayer is a group address chosen to limit the propagation of those frames to the desired peer. Appropriate addresses include the C-VLAN component reserved addresses (see Table 8-1 in IEEE Std 802.1Q-2018) and the S-VLAN component reserved addresses (see Table 8-2 in IEEE Std 802.1Q-2018). The choice of address used determines the scope of propagation of Link Aggregation control and marker PDUs within a Bridged LAN, as follows:

a) The Nearest Customer Bridge group address is an address that no conformant C-VLAN component or MAC Bridge forwards. However, this address is relayed by Two-Port MAC Relay (TPMR) components and S-VLAN components. Therefore, Link Aggregation control, Distributed Relay

control, and marker PDUs received on this address represent information about stations that are not separated from the recipient station by any intervening C-VLAN components or MAC Bridges. There can, however, be TPMR components and/or S-VLAN components in the path between the originating and receiving stations.

NOTE 1—This address makes it possible to communicate information between end stations and/or Customer Bridges and for that communication to be transparent to the presence or absence of TPMRs or S-VLAN components in the communication path. The scope of this address is the same as that of a Customer-Bridge-to-Customer-Bridge MACsec connection.

b) The Slow_Protocols_Multicast group address is an address that no conformant TPMR component, S-VLAN component, C-VLAN component, or MAC Bridge can forward. Link Aggregation control, Distributed Relay control, and marker PDUs transmitted using this DA can therefore travel no further than the stations that can be reached via a single individual LAN from the originating station. Link Aggregation control, Distributed Relay control, and marker PDUs received on this address therefore represent information about stations that are attached to the same individual LAN segment as the recipient station.

NOTE 2—This address makes it possible to transmit a Link Aggregation Control, DRCP, or Marker frame containing information specific to a single individual LAN, and for that information not to be propagated further than the extent of that individual LAN.

c) The Nearest non-TPMR Bridge group MAC address is an address that no conformant C-VLAN component, S-VLAN component, or MAC Bridge can forward. However, this address is relayed by TPMR components. Therefore, Link Aggregation control, Distributed Relay control, and marker PDUs received on this address represent information about stations that are not separated from the recipient station by any intervening C-VLAN component, S-VLAN component, or MAC Bridge. There can, however, be one or more TPMR components in the path between the originating and receiving stations.

NOTE 3—This address makes it possible to communicate information between end stations and/or Bridges and for that communication to be transparent to the presence or absence of TPMRs in the transmission path. This address is primarily intended for use within Provider Bridged Networks.

The Slow_Protocols_Multicast group address is the default for LACP and Marker Protocol destination address. The use and encoding of the Slow_Protocols_Multicast group address is specified in Annex 57A of IEEE Std 802.3-2018.

NOTE 4—IEEE Std 802.1AX-2014 and IEEE Std 802.1AXbk-2012 restricted the LACP and Marker protocol destination addresses to those in Table 6-1. This restriction has been removed, thus permitting the use of other suitable address scopes without further revision of this standard.

**Table 6-1—Link Aggregation protocol destination addresses**

| Assignment | Value |
|---|---|
| Nearest Customer Bridge group address | 01-80-C2-00-00-00 |
| IEEE 802.3 Slow_Protocols_Multicast group address | 01-80-C2-00-00-02 |
| Nearest non-TPMR Bridge group address | 01-80-C2-00-00-03 |

## 6.3 Link Aggregation Control

Link Aggregation Control uses locally configured information and information exchanged by the Link Aggregation Control Protocol.

For each Aggregation Port in the System, Link Aggregation Control

a) Maintains configuration information (reflecting the inherent properties of the individual links as well as those established by management) to control aggregation.
b) Exchanges configuration information with other Systems to allocate the link to a LAG.

NOTE—A given link is allocated to, at most, one LAG at a time. The allocation mechanism attempts to maximize aggregation, subject to management controls.

c) Attaches the Aggregation Port to the Aggregator used by the LAG, and detaches the Aggregation Port from the Aggregator when it is no longer used by the LAG.
d) Enables and disables frame collection from the Aggregation Port by the Aggregator and distribution to the Aggregation Port by the Aggregator.

Link Aggregation Control uses the following identifiers to determine whether links provide connectivity to the same peer System, to determine whether the capabilities of the Aggregation Ports attached to those links are compatible with each other and with an available Aggregator, and to support management of LAGs and their associated Aggregators:

e) A globally unique identifier for each System that participates in Link Aggregation (see 6.3.2).
f) A locally unique identifier labeling a set of compatible capabilities.
g) A means of identifying a LAG and its associated Aggregator.

System identification allows the detection of links that are connected in a loopback configuration (i.e., both ends of the same link are connected to the same System).

### 6.3.1 Characteristics of Link Aggregation Control

Link Aggregation Control provides a configuration capability that is the following:

a) **Automatic.** In the absence of manual override controls, an appropriate set of LAGs is automatically configured, and individual links are allocated to those LAGs. If a set of links can aggregate, they do aggregate.
b) **Continuous.** Manual intervention or initialization events are not a requirement for correct operation. The configuration mechanism continuously monitors for changes in state that require reconfiguration. The configuration functions detect and correct misconfigurations by performing reconfiguration and/or by taking misconfigured links out of service.
c) **Deterministic.** The configuration can resolve deterministically; i.e., the configuration achieved can be made independent of the order in which events occur and be completely determined by the combination of the capabilities of the individual links and their physical connectivity.
d) **Controllable.** The configuration capabilities accommodate devices with differing hardware and software constraints on Link Aggregation.
e) **Compatible.** Links that cannot take part in Link Aggregation, either because of their inherent capabilities or of the capabilities of the devices to which they attach, operate as normal IEEE 802 links. The introduction of Link Aggregation capability at one or both ends of a link does not result in a degradation of the perceived performance of the link.
f) **Rapid.** The configuration resolves rapidly to a stable configuration. Convergence can be achieved by the exchange of three LACPDUs, without dependence on timer values.

With the following:

g) **Low risk of misdelivery.** The operation of the (re)configuration functions minimizes the risk of frames being delivered to the wrong Aggregator.

h) **Low risk of duplication or misordering.** The operation of the (re)configuration functions minimizes the risk of frame duplication and frame misordering.

i) **Low protocol overhead.** The overhead involved in external communication of configuration information between devices is small.

### 6.3.2 System identification

The System Identifier is the concatenation of the System Priority and the System MAC address. The MAC Address needs to be globally unique or at least unique among any set of Aggregation Systems with potential links to each other; otherwise, LACP can fail to create aggregations and/or can make improper connections between systems. The MAC address chosen can be the individual MAC address associated with one of the Aggregation Ports of the System. For DRNI (Clause 9), each DRNI System in a DRNI has a unique System Identifier, but the Aggregator Ports supporting the DRNI Gateway in each DRNI System have the same System Identifier, which is provided by the concatenation of the DRNI Aggregator Port's MAC address and the DRNI Aggregator Port's System Priority.

Where it is necessary to perform numerical comparisons between System Identifiers, each System Identifier is considered to be an eight-octet unsigned binary number, constructed as follows:

a) The two most significant octets of the System Identifier comprise the System Priority. The System Priority value is taken to be an unsigned binary number; the most significant octet of the System Priority forms the most significant octet of the System Identifier.

b) The third most significant octet of the System Identifier is derived from the initial octet of the MAC address; the least significant bit of the octet is assigned the value of the first bit of the MAC address, the next most significant bit of the octet is assigned the value of the next bit of the MAC address, and so on. The fourth through eighth octets are similarly assigned the second through sixth octets of the MAC address.

### 6.3.3 Aggregator identification

Each Aggregator to which one or more Aggregation Ports are attached shall be assigned a unique, globally or locally administered individual MAC address. The MAC address assigned to the Aggregator can be the same as the MAC address assigned to one of its attached Aggregation Ports. No Aggregator shall be assigned a MAC address that is the same as that of an Aggregation Port attached to a different Aggregator within the System. When receiving frames, an Aggregation Port is never required to recognize more than one unicast address, i.e., the Aggregator's MAC address.

When management is supported, an Aggregator shall be assigned an integer identifier that uniquely identifies the Aggregator within the System. This value will typically be the same as the interface identifier (ifIndex) used for the MIB (D.4.1).

### 6.3.4 Port identification

Link Aggregation Control uses a Port Identifier, comprising the concatenation of a Port Priority (7.3.2.1.15) and a Port Number (7.3.2.1.14), to identify an Aggregation Port. Port Numbers (and hence, Port Identifiers) shall be uniquely assigned within a System. Port Number 0 shall not be assigned to any Aggregation Port.

When it is necessary to perform numerical comparisons between Port Identifiers, each Port Identifier is considered to be a four-octet unsigned binary number constructed as follows:

a) The most significant and second most significant octets are the first and second most significant octets of the Port Priority, respectively.

b) The third and fourth most significant octets are the first and second most significant octets of the Port Number, respectively.

### 6.3.5 Capability identification

The ability of one Aggregation Port to aggregate with another is summarized by a simple integer parameter, known as a Key. This facilitates communication and comparison of aggregation capabilities, which are determined by a number of factors, including the following:

a) The Aggregation Port's physical characteristics, such as point-to-point or shared medium.
b) Configuration constraints established by the network administrator.
c) Use of the Aggregation Port by higher layer protocols (e.g., assignment of Network Layer addresses).
d) Characteristics or limitations of the Aggregation Port implementation itself.

Each Aggregation Port has an operational Key and an administrative Key. The operational Key is the Key that is currently in active use for the purposes of forming aggregations. The administrative Key allows manipulation of Key values by management. The administrative and operational Keys assigned to an Aggregation Port can differ under the following circumstances:

e) When the operation of the implementation is such that an administrative change to a Key cannot be immediately reflected in the operational state of the Aggregation Port.
f) When the System supports the dynamic manipulation of Keys, as discussed in 6.7.2, either to accurately reflect changes in operational capabilities of the Aggregation Port (for example, as a result of Auto-Negotiation) or to provide a means of handling constraints on aggregation capability.

A given Key is meaningful only in the context of the System that allocates it; there is no global significance to Key values. Similarly, the relationship between administrative and operational Keys is meaningful only in the context of the System that allocates it. When a System assigns an administrative Key to a set of Aggregation Ports, it signifies that those Aggregation Ports have the potential to aggregate together, subject to the considerations discussed in 6.7.2. When a System assigns an operational Key to a set of Aggregation Ports, it signifies that, in the absence of other constraints, any or all of those Aggregation Ports can be aggregated together from the perspective of the System making the assignment. The set of Aggregation Ports that actually aggregate together are those that terminate at a common Partner System and for which that Partner System has assigned a common operational Key, local to that Partner.

A System can determine that a given link is not able to be aggregated with other links. Such links are referred to as Solitary links (as opposed to Aggregateable links). A System can declare a link to be Solitary if the inherent properties of the link allow its use as part of an aggregation, but the system is aware of no other links that are capable of aggregating with this link (e.g., the System has allocated a unique operational Key to the link).

The capability information communicated between Systems, therefore, includes this local knowledge of the aggregation capability of the link in addition to the operational Key, i.e., whether the System considers the link to be Aggregateable or Solitary.

Each Aggregator has an operational Key and an administrative Key. The values of administrative and operational Key for an Aggregator can differ in the same manner as that of Aggregation Port Keys, per item e) and item f) in this subclause. Aggregation Ports with a given operational Key can be attached only to an Aggregator that has the same operational Key.

All Keys are 16-bit identifiers. All values except the null value (all zeros) are available for local use.

NOTE—This model allows for two convenient initial configurations. The first is achieved by assigning each Aggregation Port an initial administrative and operational Key value identical to its Port Number, and assigning the same Port Numbers as Keys to the corresponding Aggregators for each Aggregation Port. A device with this initial

configuration will bring up all links as Solitary, non-aggregated links. The second is achieved by assigning the same administrative and operational Key values to all Aggregation Ports with a common set of capabilities, and also to all Aggregators. A device with this initial configuration will attempt to aggregate together any set of links that have the same Partner System Identifier and operational Key, and for which both Systems are prepared to allow aggregation.

### 6.3.6 Link Aggregation Group identification

A Link Aggregation Group consists of either

a) One or more Aggregateable links that terminate in the same pair of Systems (or DRNIs) and whose Aggregation Ports have the same operational Key, or
b) A Solitary link.

#### 6.3.6.1 Construction of the Link Aggregation Group Identifier

A unique Link Aggregation Group Identifier (LAG ID) is constructed from the following parameters for each of the communicating Systems:

a) The operational System Identifier
b) The operational Key assigned to the Aggregation Ports in the LAG
c) The Port Identifier, if the link is identified as a Solitary link

A compound identifier formed from the System Identifiers and Keys alone is sufficient to identify a LAG comprising Aggregateable links. However, such an identifier is not sufficient for a LAG comprising a single Solitary link where the Partner System Identifier and operational Key can be zero. Even if these are nonzero, there can be multiple Solitary links with the same System Identifier and operational Key combinations, and it is necessary to include Port Identifiers to provide unique LAG IDs.

Given that

d) S and T are System Identifiers,
e) K and L are the operational Keys assigned to a LAG by S and T, respectively, and
f) P and Q are the Port Identifiers of the Aggregation Ports being attached if the LAG comprises a single Solitary link and zero if the LAG comprises one or more Aggregateable links,

then the general form of the unique LAG ID is [(SKP), (TLQ)].

To simplify comparison of LAG IDs, it is conventional to order these so that S is the numerically smaller of S and T.

#### 6.3.6.2 Representation of the Link Aggregation Group Identifier

To allow for convenient transcription and interpretation by human network personnel, this standard provides a convention for representing LAG IDs. Using this format,

a) All fields are written as hexadecimal numbers, two digits per octet, in canonical format.
b) Octets are presented in order, from left to right. Within fields carrying numerical significance (e.g., priority values), the most significant octet is presented first, and the least significant octet last.
c) Within fields that carry MAC addresses, organizationally unique identifiers (OUIs) or Company Identifiers (CIDs), successive octets are separated by hyphens (-), in accordance with the hexadecimal representation for MAC addresses defined in IEEE Std 802.
d) Parameters of the LAG ID are separated by commas.

For example, consider a link connecting two Aggregation Ports with the parameters shown in Table 6-2.

**Table 6-2—Example Partner parameters**

| | Partner SKP | Partner TLQ |
|---|---|---|
| System Parameters (S, T) | System Priority = 0x8000 (see 6.4.2.3)<br>System MACaddr= AC-DE-48-03-67-80 | System Priority = 0x8000 (see 6.4.2.3)<br>System MACaddr = AC-DE-48-03-FF-FF |
| Key Parameter (K, L) | Key = 0x0001 | Key = 0x00AA |
| Aggregation Port Parameters (P, Q) | Port Priority = 0x80 (see 6.4.2.3)<br>Port Number = 0x0002 | Port Priority = 0x80 (see 6.4.2.3)<br>Port Number = 0x0002 |

For a Solitary link, which cannot be aggregated with other links, the LAG ID derived from this information is represented as follows:

[(SKP), (TLQ)] = [(8000,AC-DE-48-03-67-80,0001,80,0002), (8000,AC-DE-48-03-FF-FF,00AA,80,0002)]

For a link that is Aggregateable, the LAG ID's Port Identifier components are zero, and the LAG ID derived from this information is represented as follows:

[(SKP), (TLQ)] = [(8000,AC-DE-48-03-67-80,0001,00,0000), (8000,AC-DE-48-03-FF-FF,00AA,00,0000)]

### 6.3.7 Selecting a Link Aggregation Group

Each Aggregation Port is selected for membership in the LAG uniquely identified by the LAG ID (composed of operational information, both derived from local administrative parameters and received through the Link Aggregation Control Protocol). Initial determination of the LAG ID is delayed to allow receipt of information from a peer Link Aggregation Control entity; in the event such information is not received, locally configured administrative defaults are assumed for the remote Aggregation Port's operational parameters.

Where a particular link is known to be Solitary, the complete LAG ID is not required to select the LAG since the link will not be aggregated with any other.

### 6.3.8 Agreeing on a Link Aggregation Group

Before frames are collected from and distributed to a link, both the local Link Aggregation Control entity and its remote peer (if present) need to agree on the LAG. The Link Aggregation Control Protocol allows each of the communicating entities to check its peer's current understanding of the LAG ID and facilitates rapid exchange of operational parameters while that understanding differs from its own.

The ability of LACP to signal that a particular link is Solitary (by clearing the Aggregation bit in the LACPDU *Actor_State* field; see 6.4.2.3) can accelerate the use of the link since, if both Link Aggregation Control entities know that the link is Solitary, agreement on the LAG ID is not necessary.

### 6.3.9 Attaching a link to an Aggregator

Once a link has selected a LAG, Link Aggregation Control can attach that link to a compatible Aggregator. An Aggregator is compatible if

a) The Aggregator's operational System Identifier matches the Aggregation Port's operational System Identifier, and
b) The Aggregator's operational Key matches the Aggregation Port's operational Key, and
c) All other links currently attached to the Aggregator have selected the same LAG.

If several compatible Aggregators exist, Link Aggregation Control can employ a locally determined selection algorithm, either to provide deterministic behavior (i.e., independence from the order in which Aggregators become available) or to maximize availability of the aggregation to an Aggregator Client. If no compatible Aggregator exists, then it is not possible to enable the link until a compatible Aggregator becomes available.

NOTE—The unavailability of a selected Aggregator can be a temporary condition encountered while links are moved between Aggregators. However, given the flexibility of the Key scheme and given that in some implementations there might not be enough Aggregators to service a given configuration of links, it is possible to create configurations in which no Aggregator is available to serve a newly identified LAG. In such cases, the links that are members of that LAG cannot become active until the configuration is changed to free up an appropriate Aggregator.

Links that are not successful candidates for aggregation (e.g., links that are attached to other devices that cannot perform aggregation or links that have been manually configured to be non-aggregateable) operate as Solitary links. For consistency of modeling, such a link is regarded as being attached to a compatible Aggregator that can be associated only with a single link. In other words, from the perspective of Link Aggregation, non-aggregated links are not a special case; they compose an aggregation with a maximum membership of one link.

More than one link can select the same LAG within a short period of time and, as these links detach from their prior Aggregators, additional compatible Aggregators can become available. In order to avoid such events causing repeated configuration changes, Link Aggregation Control can delay reattachment and allows multiple links to be attached to an Aggregator at the same time.

### 6.3.10 Signaling readiness to transfer user data

Once a link has been attached to an Aggregator (6.3.9) compatible with the agreed-upon LAG (6.3.8), each Link Aggregation Control entity signals to its peer its readiness to transfer user data to and from the Aggregator's Aggregator Client. In addition to allowing time for the organization of local Aggregator resources, including the possibility that a compatible Aggregator does not exist, explicit signaling of readiness to transfer user data can be delayed to ensure preservation of frame ordering and prevention of frame duplication. Link Aggregation Control does not signal readiness until it is certain that there are no frames in transit on the link that were transmitted while the link was a member of a previous LAG. This can involve the use of an explicit Marker protocol that ensures that no frames remain to be received at either end of the link before reconfiguration takes place. The operation of the Marker protocol is described in 6.5. The decision about when, or if, the Marker protocol is used is entirely dependent upon the nature of the distribution algorithm that is employed.

### 6.3.11 Enabling the Frame Collector and Frame Distributor

Initially, both the Frame Collector and Frame Distributor are disabled. Once the Link Aggregation Control entity is ready to transfer user data using the link and its peer entity has also signaled readiness, the process of enabling the link can proceed. When any Aggregation Port attached to the Frame Collection function is Collecting, the Frame Collector is enabled (thus preparing it to receive frames sent over the link by the

remote Aggregator's Distributor) and that fact is communicated to the Partner. Once the received information indicates that the remote Aggregator's Frame Collector is enabled, the Frame Distributor is also enabled.

NOTE—This description assumes that the implementation is capable of controlling the state of the transmit and receive functions of the MAC independently. In an implementation where this is not possible, the transmit and receive functions are enabled or disabled together. The manner in which this is achieved is detailed in the description of the Mux machine (see 6.4.13).

### 6.3.12 MAC_Operational status

The ISS MAC_Operational status parameter (11.2 of IEEE Std 802.1AC-2016) indicates to a higher layer entity that a Service Access Point is (TRUE) or is not (FALSE) available for use. A physical link Service Access Point uses MAC_Operational to provide link status to the Link Aggregation Sublayer, and each Aggregator uses MAC_Operational to provide LAG connectivity status to its attached Client (Figure 6-3).

MAC_Operational from the service layer underlying an Aggregation Port controls the Port_Operational variable (6.4.6). Through this variable, collecting and distributing are disabled on a link whose MAC_Operational status is FALSE.

The Link Aggregation Sublayer shall present its MAC_Operational status as TRUE to higher layers if the operational state of the Aggregator is "up" (Aggregator_Operational (6.4.5) is TRUE) to indicate that one or more of the Aggregation Ports attached to the Aggregator are both Collecting and Distributing. If none of the Aggregation Ports that are attached to the Aggregator are Collecting and Distributing, or if there are no Aggregation Ports attached to this Aggregator, then the operational state is "down" (Aggregator_Operational is FALSE). The operational state of the Aggregator is reported by the aAggOperState (7.3.1.1.14) managed object.

### 6.3.13 Monitoring the membership of a Link Aggregation Group

Each link is monitored in order to confirm that the Link Aggregation Control functions at each end of the link still agree on the configuration information for that link. If the LAG information changes for the link or for another link attached to the same Aggregator, it can be necessary to detach the link's Aggregation Port from its current Aggregator and select another Aggregator.

### 6.3.14 Detaching a link from an Aggregator

If a change in LAG membership requires the selection of a different Aggregator for a link, then the Collecting and Distributing states for that link's Aggregation Port are set to FALSE to tell the Frame Collection and Distribution functions to stop distributing to and collecting from the link. Then the Aggregation Port is detached from the Aggregator. LACP communicates those changes to the Link Aggregation Partner.

Once a link's Aggregation Port has been detached from an Aggregator, another Aggregator with the same operational Key as that for the Aggregation Port (and no attached Aggregation Ports for links in a different LAG) can be selected, and the Aggregation Port can be attached to that Aggregator.

Link Aggregation Control can avoid reordering frames when a conversation is reallocated to a different link. The Marker protocol (6.5) can be used to check that no frames that form part of that conversation remain to be received at either end of the old link before the conversation can proceed on the new link. Alternatively, Conversation-Sensitive Collection and Distribution (6.6) can be used with Discard Wrong Conversation (DWC) to control which link is to be used to receive frames for that conversation.

### 6.3.15 Configuration and administrative control of Link Aggregation

The way that links are aggregated can be controlled through management. In particular, administrative configuration allows

a) Key values associated with an Aggregation Port to be read or changed.
b) Key values associated with an Aggregator to be read or changed.
c) Links to be identified as being incapable of aggregation.
d) Link Aggregation Control Protocol parameters to be read or changed.

### 6.3.16 Link Aggregation Control state information

The Link Aggregation Control function maintains the following information for each link:

a) The identifier of the LAG to which it currently belongs.
b) The identifier of the Aggregator associated with that LAG.
c) The status of interaction between the Frame Collection function of the Aggregator and the link (Collecting TRUE or Collecting FALSE). Collecting TRUE indicates that the receive function of this link is enabled with respect to its participation in an aggregation; i.e., received frames are passed up to the Aggregator for collection.
d) The status of interaction between the Frame Distribution function of the Aggregator and the link (Distributing TRUE or Distributing FALSE). Distributing TRUE indicates that the transmit function of this link is enabled with respect to its participation in an aggregation; i.e., frames can be passed down from the Aggregator's Frame Distribution function for transmission.

This state information is communicated directly between Link Aggregation Control and the Aggregator through shared state variables without the use of a formal service interface.

The Link Aggregation Control function maintains the following information with respect to each Aggregator:

e) The operational status of the Aggregator, indicating whether the Aggregator is available for use by the Aggregator Client.
f) A list of all Aggregation Ports that have currently attached to, or have selected, the Aggregator.

## 6.4 Link Aggregation Control Protocol

The Link Aggregation Control Protocol (LACP) provides a standardized means for exchanging information between Partner Systems on a link to allow their Link Aggregation Control instances to identify the LAG to which the link belongs, add the link to that LAG, and enable its transmission and reception functions in an orderly manner.

### 6.4.1 LACP design elements

The following considerations were taken into account during the development of LACP:

a) The protocol depends upon the transmission of information and state, rather than the transmission of commands. LACPDUs sent by the first party (the Actor) convey to the second party (the Actor's protocol Partner) what the Actor knows about both its own state and that of the Partner.
b) The information conveyed in the protocol is sufficient to allow the Partner to determine what action to take next.

c) Active or passive participation in LACP is controlled by LACP_Activity, an administrative control associated with each Aggregation Port, that can take the value Active LACP or Passive LACP. Passive LACP indicates the Aggregation Port's preference for not transmitting LACPDUs unless its Partner's control value is Active LACP (i.e., a preference not to speak unless spoken to). Active LACP indicates the Aggregation Port's preference to participate in the protocol regardless of the Partner's control value (i.e., a preference to speak regardless).

d) Periodic transmission of LACPDUs occurs if the LACP_Activity control of either the Actor or the Partner is Active LACP. These periodic transmissions occur at either a slow or fast transmission rate depending upon the expressed preference (Long Timeout or Short Timeout) of the Partner System.

e) In addition to periodic LACPDU transmissions, the protocol transmits LACPDUs when there is a Need To Transmit (NTT) something to the Partner, i.e., when the Actor's state changes or when it is apparent from the Partner's LACPDUs that the Partner does not know the Actor's current state.

f) The protocol assumes that the rate of LACPDU loss is very low.

If information is received from the Partner indicating that it does not have up-to-date information on the Actor's state or if the next periodic transmission is due, then the Actor transmits an LACPDU that will update the Partner.

### 6.4.2 LACPDU structure and encoding

#### 6.4.2.1 Transmission and representation of octets

All LACPDUs comprise an integral number of octets. The octets in an LACPDU are numbered starting from 1 and increasing in the order they are put into the LACPDU. The bits in each octet are numbered from 1 to 8, where bit 1 is the least significant bit.

When consecutive bits within an octet are used to represent a binary number, the highest bit number has the most significant value. When consecutive octets are used to represent a binary number, the most significant octet has the lowest octet number and is transmitted first, followed by successively less significant octets (with successively higher octet numbers).

When the encoding of (an element of) an LACPDU is depicted in a diagram,

a) Elements of an LACPDU are transmitted from top to bottom.

b) When an element contains multiple octets, the most significant octet is transmitted first.

c) When multiple octets are depicted side by side, the octets are transmitted from left to right.

d) Within an octet, bits are shown with bit 8 (the most significant bit) to the left and bit 1 (the least significant bit) to the right.

#### 6.4.2.2 Encapsulation of LACPDUs in frames

An LACPDU is encoded in the mac_service_data_unit parameter of an M_UNITDATA.request or M_UNITDATA.indication. The first octets of the mac_service_data_unit are the Slow Protocols EtherType (Table 6-3), followed by the LACPDU as specified in 6.4.2.3.

**Table 6-3—Slow Protocols EtherType Assignment**

| Assignment | Value |
|---|---|
| Slow Protocols EtherType (specified in Annex 57A of IEEE Std 802.3-2018) | 88–09 |

### 6.4.2.3 LACPDU structure

The LACPDU structure, as generated by the LACP Transmit machine (6.4.13), is shown in Figure 6-8 and further described in the field definitions after the figure. A Version 1 LACPDU does not include any "Other TLVs". The structure and field definitions of other TLVs that can be included in Version 2 LACPDUs are described in 6.4.2.4. Requirements regarding the validation and processing of received LACPDUs are specified in the recordPDU function of the LACP Receive machine (6.4.11).

| | Length (in octets) |
|---|---|
| **Subtype = LACP** | **1** |
| **Version Number** | **1** |
| **TLV_Type = Actor Information** | **1** |
| Actor Information Length = 20 | 1 |
| Actor_System_Priority | 2 |
| Actor_System | 6 |
| Actor_Key | 2 |
| Actor_Port_Priority | 2 |
| Actor_Port | 2 |
| Actor_State | **1** |
| Reserved | 3 |
| **TLV_Type = Partner Information** | 1 |
| Partner Information Length = 20 | 1 |
| Partner_System_Priority | **2** |
| Partner_System | 6 |
| Partner_Key | 2 |
| Partner_Port_Priority | 2 |
| Partner_Port | 2 |
| Partner_State | 1 |
| Reserved | 3 |
| **TLV_Type = Collector Information** | 1 |
| Collector Information Length = 16 | **1** |
| CollectorMaxDelay | **2** |
| Reserved | **12** |
| **Other TLVs** | **L** |
| **TLV_Type = Terminator** | **1** |
| Terminator Length = 0 | **1** |
| **Pad** | **50-L** |

**Figure 6-8—LACPDU structure**

a) *Subtype.* The Subtype field identifies the specific Slow Protocol being encapsulated. LACPDUs carry the Subtype value 0x01.

b) *Version Number.* This identifies the LACP version as contained in Actor_System_LACP_Version.

c) *TLV_Type = Actor Information.* This field indicates the nature of the information carried in this TLV-tuple. Actor information is identified by the value 0x01.

d) *Actor Information Length.* This field indicates the length (in octets) of this TLV-tuple. Actor information uses a length value of 20 (0x14).

e) *Actor_System_Priority.* The operational value of the priority component of the System Identifier assigned by the Actor for the Aggregation Port transmitting the LACPDU.

f) *Actor_System.* The operational value of the MAC address component of the System Identifier assigned by the Actor for the Aggregation Port transmitting the LACPDU.

g) *Actor_Key.* The operational Key value assigned to the Aggregation Port by the Actor.

h) *Actor_Port_Priority.* The priority assigned to this Aggregation Port by the Actor (the System sending the PDU; assigned by management or administration policy).

i) *Actor_Port.* The Port Number assigned to the Aggregation Port by the Actor (the System sending the PDU).

j) *Actor_State.* The Actor's state variables for the Aggregation Port, encoded as individual bits within a single octet, as illustrated in Figure 6-9 and described after the figure.

| BIT<br>8 (MSB) | 7 | 6 | 5 | 4 | 3 | 2 | 1 (LSB) |
|---|---|---|---|---|---|---|---|
| Expired | Defaulted | Distributing | Collecting | Synchronization | Aggregation | Short_Timeout | LACP_Activity |

**Figure 6-9—Bit encoding of the Actor_State and Partner_State fields**

NOTE 1—Previous versions of this standard included a diagram of the Actor_State and Partner_State bit encoding with bits numbered from 0 (LSB) to 7 (MSB) from left to right. This diagram has been modified to conform to IEEE 802.1 conventions of numbering bits from 8 (MSB) to 1 (LSB) from left to right. This change affects only how the fields are diagrammed; the bit encoding as transmitted remains the same.

1) *LACP_Activity* is encoded in bit 1. This flag indicates the Activity control value with regard to this link. Active LACP is encoded as a 1; Passive LACP is encoded as a 0.
2) *Short_Timeout* is encoded in bit 2. This flag indicates the Timeout control value with regard to this link. Short Timeout is encoded as a 1; Long Timeout is encoded as a 0.
3) *Aggregation* is encoded in bit 3. If TRUE (encoded as a 1), this flag indicates that the System considers this link to be *Aggregateable*, i.e., a potential candidate for aggregation. If FALSE (encoded as a 0), the link is considered to be *Solitary*; i.e., this link can be operated only as a Solitary link.
4) *Synchronization* is encoded in bit 4. If TRUE (encoded as a 1), the System considers this link to have been allocated to the correct LAG, the group has been associated with a compatible Aggregator, and the identity of the LAG is consistent with the System Identifier and operational Key information transmitted. If FALSE (encoded as a 0), then this link is currently not in the right LAG.
5) *Collecting* is encoded in bit 5. TRUE (encoded as a 1) means collection of incoming frames on this link is definitely enabled; i.e., collection is currently enabled and is not expected to be disabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise FALSE (encoded as a 0).
6) *Distributing* is encoded in bit 6. FALSE (encoded as a 0) means distribution of outgoing frames on this link is definitely disabled; i.e., distribution is currently disabled and is not expected to be enabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise TRUE (encoded as a 1).
7) *Defaulted* is encoded in bit 7. If TRUE (encoded as a 1), this flag indicates that the Actor's LACP Receive machine is using Defaulted operational Partner information, administratively configured for the Partner. If FALSE (encoded as a 0), the operational Partner information in use has been received in an LACPDU.

8) *Expired* is encoded in bit 8. If TRUE (encoded as a 1), this flag indicates that the Actor's LACP Receive machine is in the EXPIRED state; if FALSE (encoded as a 0), this flag indicates that the Actor's LACP Receive machine is not in the EXPIRED state.

NOTE 2—The received values of Defaulted and Expired state are not used by LACP; however, knowing their values can be useful when diagnosing protocol problems.

k) *Reserved.* These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zero.

l) *TLV_Type = Partner Information.* This field indicates the nature of the information carried in this TLV-tuple. Partner information is identified by the integer value 0x02.

m) *Partner Information Length.* This field indicates the length (in octets) of this TLV-tuple. Partner information uses a length value of 20 (0x14).

n) *Partner_System_Priority.* The priority assigned to the Partner System (by management or administration policy).

o) *Partner_System.* The MAC address component of the Partner's System Identifier.

p) *Partner_Key.* The operational Key value assigned to the Aggregation Port associated with this link by the Partner.

q) *Partner_Port_Priority.* The priority assigned to this Aggregation Port by the Partner (by management or administration policy).

r) *Partner_Port.* The Port Number associated with this link assigned to the Aggregation Port by the Partner.

s) *Partner_State.* The Actor's view of the Partner's state variables, depicted in Figure 6-9 and encoded as individual bits within a single octet, as defined for Actor_State.

t) *Reserved.* These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zero.

u) *TLV_Type = Collector Information.* This field indicates the nature of the information carried in this TLV-tuple. Collector information is identified by the integer value 0x03.

v) *Collector Information Length.* This field indicates the length (in octets) of this TLV-tuple. Collector information uses a length value of 16 (0x10).

w) *CollectorMaxDelay.* This field contains the value of CollectorMaxDelay (6.2.3.1.1) of the station transmitting the LACPDU in tens of microseconds. The range of values for this parameter is 0 to 65 535 tens of microseconds (0.65535 s).

x) *Reserved.* These 12 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zero.

y) *Other TLVs.* This field contains additional TLVs (Version 2 TLVs are listed in 6.4.2.4). The length L equals the sum of the lengths of all individual additional TLVs.

z) *TLV_Type = Terminator.* This field indicates the nature of the information carried in this TLV-tuple. Terminator (end of message) information is identified by the integer value 0x00.

aa) *Terminator Length.* This field indicates the length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).

NOTE 3—The use of a Terminator Length of 0 is an intentional exception to the defined TLV_Length encoding. In TLV encoding schemes, it is common practice for the terminator encoding to be 0 for both the type and the length.

ab) *Pad*. These 50-*L* octets are ignored on receipt and are transmitted as zeros to claim compliance with this standard.

NOTE 4—Previous versions of this standard (IEEE Std 802.1AX-2008 and earlier) forced the total length of the LACPDU to result in a fixed frame size of 128 octets on IEEE 802.3 media. Although the total length of the LACPDU was not part of the validation of received LACPDUs, it is possible that some Version 1 implementations accept only LACPDUs with this length. Therefore, this version of the standard maintains the practice of padding LACPDUs to a fixed length.

#### 6.4.2.4 Version 2 TLVs

Table 6-4 provides a list of all TLVs that are applicable for Version 2 LACP. They support the Conversation-Sensitive Collection and Distribution functionality described in 6.6.

**Table 6-4—Type field values of Version 2 TLVs**

| TLV | Type field |
|---|---|
| Port Algorithm TLV | 0x04 |
| Port Conversation Link Digest TLV | 0x05 |
| Deprecated | 0x06 |
| Deprecated | 0x07 |
| Deprecated | 0x08 |
| Deprecated | 0x09 |
| Port Conversation Service Digest TLV | 0x0A |

A TLV with a value in the Type field that has been deprecated is not included in a transmitted LACPDU and is ignored if present in a received LACPDU.

##### 6.4.2.4.1 Port Algorithm TLV

This TLV supports Conversation-sensitive LACP operation (6.6.3) and shall be included in any implementation supporting CSCD. The Port Algorithm TLV structure is shown in Figure 6-10 and further described in the field definitions after the figure.

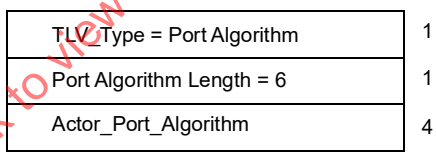



**Figure 6-10—Port Algorithm TLV**

a) *TLV_Type = Port Algorithm.* The Port Algorithm TLV is identified by the integer value 0x04.

b) *Port Algorithm Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Algorithm TLV uses a length value of 6 (0x06).

c) *Actor_Port_Algorithm*. This field contains the value of the Actor_Oper_Port_Algorithm used to assign frames to Port Conversation IDs. It comprises the 3-octet OUI or CID identifying the organization that is responsible for this distribution algorithm and one following octet used to identify the Port Algorithm specified by that organization. The most significant bit of that fourth octet indicates whether the algorithm makes use of the Admin_Conv_Service_Map. Table 8-1 provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings.

#### 6.4.2.4.2 Port Conversation Link Digest TLV

This TLV is required when the implementation supports CSCD and the Actor_Port_Algorithm is anything other than "Unspecified" (Table 8-1). The Port Conversation Link Digest TLV structure is shown in Figure 6-11 and further described in the field definitions after the figure.

| | |
|---|---|
| TLV_Type = Port Conversation Link Digest | 1 |
| Port Conversation Link Digest Length = 20 | 1 |
| Link_Number | 2 |
| Actor_Conv_Link_Digest | 16 |

**Figure 6-11—Port Conversation Link Digest TLV**

a) *TLV_Type = Port Conversation Link Digest.* The Port Conversation Link Digest TLV is identified by the integer value 0x05.

b) *Port Conversation Link Digest Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Conversation Link Digest TLV uses a length value of 2 (0x14).

c) *Link_Number.* This field contains the operational value of the Link_Number assigned to this Aggregation Port.

d) *Actor_Conv_Link_Digest.* This field contains the value of the MD5 digest Actor_Oper_Conv_Link_Digest (see IETF RFC 1321).

#### 6.4.2.4.3 Port Conversation Service Digest TLV

This TLV is required when the implementation supports CSCD and the most significant bit of the fourth octet of the Actor_Port_Algorithm is 1, indicating that algorithm makes use of the Admin_Conv_Service_Map to map Service IDs to Conversation IDs. The Port Conversation Service Digest TLV structure is shown in Figure 6-12 and further described in the field definitions after the figure.

| | |
|---|---|
| TLV_Type = Port Conversation Service Digest | 1 |
| Port Conversation Service Digest Length = 18 | 1 |
| Actor_Conv_Service_Digest | 16 |

**Figure 6-12—Port Conversation Service Digest TLV**

a) *TLV_Type = Port Conversation Service Digest.* The Port Conversation Service Digest TLV is identified by the integer value 0x0A.

b) *Port Conversation Service Digest Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Conversation Service Digest TLV uses a length value of 18 (0x12).

c) *Actor_Conv_Service_Digest.* This field contains the value of the MD5 digest Actor_Oper_Conv_Service_Digest.

### 6.4.3 LACP state machine overview

Figure 6-13 is not itself a state machine but provides an overview of the LACP state machines and the variables that facilitate their operation and interactions. Each LACP state machine is represented as a box with one section containing a list of the primary variables controlled by that machine and another section containing a list of the functions invoked by that machine. Communication between machines is represented by arrows labeled with the variable being communicated. An arrow with a solid (black) arrowhead indicates that the variable is not modified by the destination machine. An arrow with a hollow (white) arrowhead indicates that the variable can be modified by the destination machine.

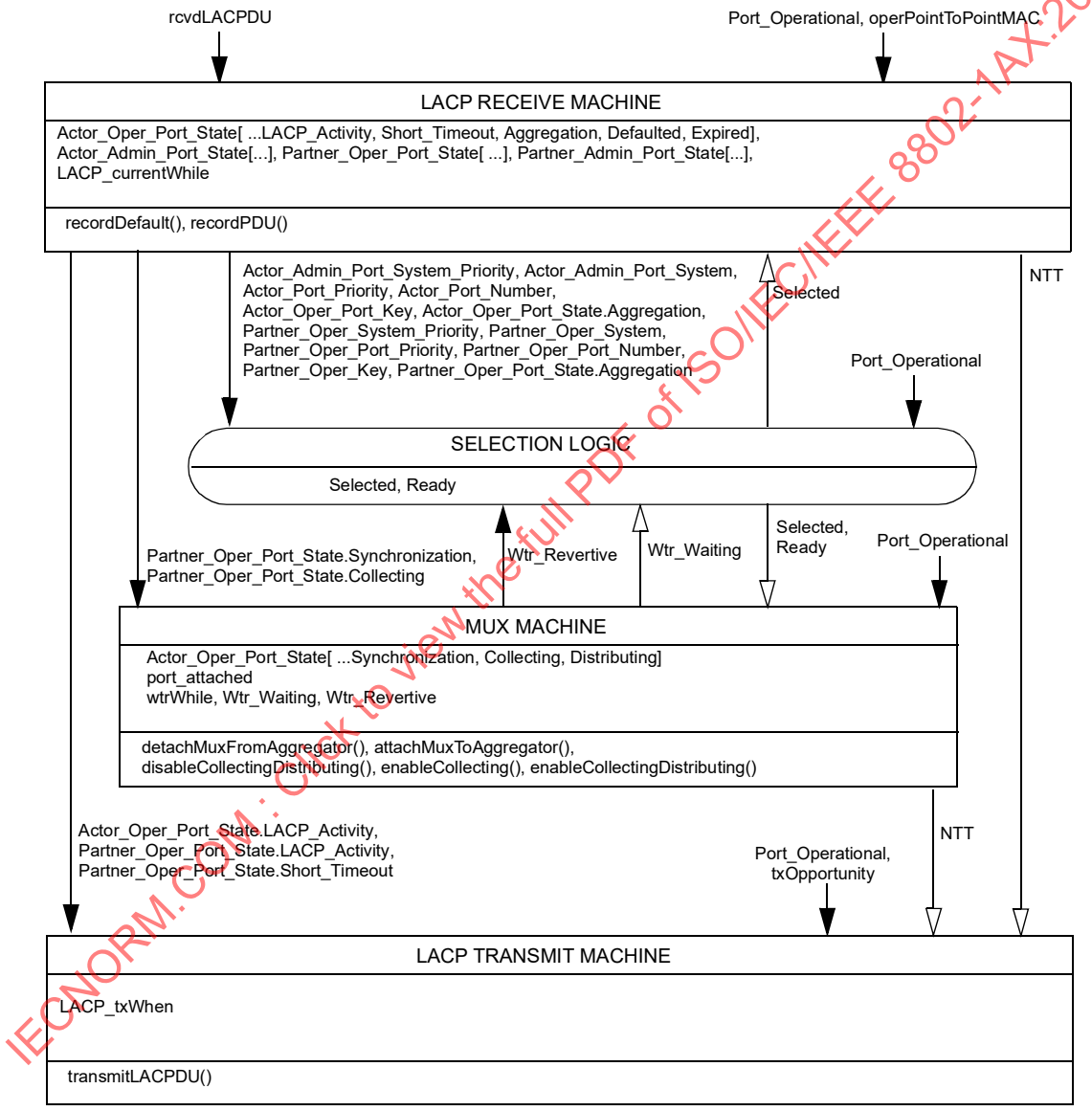



**Figure 6-13—LACP state machine overview**

There are three state machines per Aggregation Port as follows:

a) *LACP Receive machine* (6.4.11). This state machine receives LACPDUs from the Partner, records the information contained, and times it out using either Short_Timeout_Time or Long_Timeout_Time, according to the configuration of Actor_Admin_Port_State.Short_Timeout. It evaluates the incoming information from the Partner to determine whether the Actor and Partner have both agreed upon the protocol information exchanged to the extent that the Aggregation Port can now be safely used, either in an aggregation with other Aggregation Ports or as an individual Aggregation Port. If not, it asserts NTT in order to transmit fresh protocol information to the Partner. If the protocol information from the Partner times out, the LACP Receive machine installs default parameter values for use by the other state machines.

b) *Mux machine* (6.4.13). This state machine is responsible for attaching the Aggregation Port to a selected Aggregator, detaching the Aggregation Port from a deselected Aggregator, and enabling or disabling collecting from and distributing to the Aggregation Port as required by the current protocol information.

c) *LACP Transmit machine* (6.4.14). This state machine handles the transmission of LACPDUs, both on demand from the other state machines and on a periodic basis.

The protocol also makes use of the Selection Logic (6.4.12), which operates using inputs from all Aggregation Ports and Aggregators in the System. The Selection Logic is responsible for selecting the Aggregator to be associated with each Aggregation Port.

### 6.4.4 Constants

All timers specified in this subclause have an implementation tolerance of ± 250 ms.

Fast_Periodic_Time
: The number of seconds between periodic transmissions using Short Timeouts.
Data Type: Integer.
Value: 1

Slow_Periodic_Time
: The number of seconds between periodic transmissions using Long Timeouts.
Data Type: Integer.
Value: 30

Short_Timeout_Time
: The number of seconds before invalidating received LACPDU information when using Short Timeouts (3 × Fast_Periodic_Time).
Data Type: Integer.
Value: 3

Long_Timeout_Time
: The number of seconds before invalidating received LACPDU information when using Long Timeouts (3 × Slow_Periodic_Time).
Data Type: Integer.
Value: 90

Aggregate_Wait_Time
: The number of seconds to delay aggregation, to allow multiple links to aggregate simultaneously.
Data Type: Integer.
Value: 2

Actor_System_LACP_Version
: The Version number of the Actor's LACP implementation. This is reported in the aAggPortActorLacpVersion (7.3.2.1.33) managed object for each Aggregation Port.
Data Type: Integer.
Value: 2

coupled_control
Indicates to the Mux machine (6.4.13) whether the implementation is capable of enabling and disabling Collection independently from Distribution (FALSE) or whether Collection and Distribution are enabled and disabled together (TRUE).
Data Type: Boolean.

### 6.4.5 Variables associated with each Aggregator

Aggregator_Enabled
Provides administrative control over the Aggregator, and is made available to the Aggregator Client via the "MAC_Enabled" status parameter of the ISS. When FALSE ("down"), no Aggregation Ports are permitted to select this Aggregator, any Aggregation Ports that have selected this Aggregator are set to UNSELECTED, and the operational state of the Aggregator will be "down" (Aggregator_Operational will be FALSE). When TRUE ("up"), the Aggregation Ports are permitted to select this Aggregator, and the operational state of the Aggregator is determined by the state of the attached Aggregation Ports.
Data Type: Boolean.
Administratively assigned.
Alias of aAggAdminState (7.3.1.1.13).

Aggregator_Operational
Indicates the operational state of the Aggregator, and is made available to the Aggregator Client via the "MAC_Operational" status parameter of the ISS. When TRUE, the Aggregator is "up" and is available for use by the Aggregator Client; when FALSE, the Aggregator is "down" and is not available for use by the Aggregator Client.
Data Type: Boolean.
TRUE when there are one or more Aggregation Ports attached to the Aggregator that are Collecting and Distributing (i.e., both Actor_Oper_Port_State.Collecting and Actor_Oper_Port_State.Distributing are TRUE).
Alias of aAggOperState (7.3.1.1.14).

Aggregator_MAC_Address
The MAC address assigned to the Aggregator.
Data Type: 6 octets.
Administratively assigned.
Alias of aAggMACAddress (7.3.1.1.9).

Aggregator_Interface_ID
Used by management to uniquely identify this Aggregator. This is represented in the SMIv2 MIB as an ifIndex (D.4.1).
Data Type: ifIndex.
Administratively assigned.
Alias of aAggID (7.3.1.1.1).

Actor_Admin_System
The MAC address component of the System Identifier of the System containing the Aggregator. When this value is changed by management, the value of Selected variable is set to UNSELECTED for all Aggregation Ports in the LAG_Ports list for this Aggregator.
Data Type: 6 octets
Administratively assigned.
Alias of aAggActorSystemID (7.3.1.1.4).

Actor_Admin_System_Priority
The System Priority component of the System Identifier of the System containing the Aggregator. When this value is changed by management, the value of Selected variable is set to UNSELECTED for all Aggregation Ports in the LAG_Ports list for this Aggregator.
Data Type: Integer

Value: 1 to 65535.
Administratively assigned.
Alias of aAggActorSystemPriority (7.3.1.1.5).

NOTE—The variables provided for the Aggregator and the Aggregation Port allow management of the Actor MAC Address and Actor System Priority. The result of this is to permit a single item of equipment to be configured by management to contain more than one System from the point of view of the operation of Link Aggregation. This can be of particular use in the configuration of equipment that has limited aggregation capability (see 6.7), or has multiple Bridge and/or end station components, or supports a DRNI Gateway (9.5).

Actor_Oper_System
The operational value of the MAC address component of the Aggregator's System Identifier. When the Aggregator does not support a DRNI Gateway, the operational value is always the same as the administrative value.
Data Type: 6 octets
Assigned by the Actor.

Actor_Oper_System_Priority
The operational value of the priority component of the Aggregator's System Identifier. When the Aggregator does not support a DRNI Gateway, the operational value is always the same as the administrative value.
Data Type: Integer.
Value: 1 to 65535.
Assigned by the Actor.

Actor_Admin_Aggregator_Key
The administrative Key value associated with the Aggregator.
Data Type: Integer.
Value: 1 to 65535.
Administratively assigned.
Alias of aAggActorAdminKey (7.3.1.1.7).

Actor_Oper_Aggregator_Key
The operational Key value associated with the Aggregator.
Data Type: Integer.
Value: 1 to 65535.
Assigned by the Actor.
Alias of aAggActorOperKey (7.3.1.1.8).

Partner_System
The MAC address component of the System Identifier of the remote System to which the Aggregator is connected. If the Aggregator has no attached Aggregation Ports, this variable is set to 0x00-00-00-00-00-00.
Data Type: 6 octets.
Alias of aAggPartnerSystemID (7.3.1.1.10).

Partner_System_Priority
The System Priority of the remote System to which the Aggregator is connected. If the Aggregator has no attached Aggregation Ports, this variable is set to zero.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPartnerSystemPriority (7.3.1.1.11).

Partner_Oper_Aggregator_Key
The operational Key assigned to an aggregation by the remote System to which this Aggregator is connected. If the Aggregator has no attached Aggregation Ports, this variable is set to zero.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPartnerOperKey (7.3.1.1.12).

LAG_Ports
A list of the Actor_Port_Interface_ID of all Aggregation Ports that belong to the Link Aggregation Group (i.e., all Aggregation Ports that have selected, or are attached to, this Aggregator).
Data Type: List of ifIndex values.
Alias of aAggPortList (7.3.1.1.30).

### 6.4.6 Variables associated with each Aggregation Port

Actor_Port_Interface_ID
Used by management to uniquely identify this Aggregation Port. This is represented in the SMIv2 MIB as an ifIndex (D.4.1) and is the same as the ifIndex of the underlying service (typically a MAC interface; see D.5.1).
Data Type: ifIndex.
Administratively assigned.
Alias of aAggPortID (7.3.2.1.1).

NOTE 1—The Actor_Port_Interface_ID is not to be confused with the "Port Identifier" defined in 6.3.4, which is a concatenation of the Actor_Port_Priority and Actor_Port_Number.

Actor_Admin_Port_System
The MAC address component of the System Identifier of the System containing the Aggregation Port. When this value is changed by management, the value of Selected variable is set to UNSELECTED, Actor_Oper_Port_State.Synchronization is set to FALSE, and Partner_Oper_Port_State.Synchronization is set to FALSE.
Data Type: 6 octets.
Administratively assigned.
Alias of aAggPortActorSystemID (7.3.2.1.3).

Actor_Admin_Port_System_Priority
The System Priority component of the System Identifier of the System containing the Aggregation Port. When this value is changed by management, the value of Selected variable is set to UNSELECTED, Actor_Oper_Port_State.Synchronization is set to FALSE, and Partner_Oper_Port_State.Synchronization is set to FALSE.
Data Type: Integer.
Value: 1 to 65535.
Administratively assigned.
Alias of aAggPortActorSystemPriority (7.3.2.1.2).

Actor_Oper_Port_System
The operational value of the MAC address component of the Aggregation Port's System Identifier. When the Aggregator does not support a DRNI Gateway, the operational value is always the same as the administrative value.
Data Type: 6 octets.

Actor_Oper_Port_System_Priority
The operational value of the priority component of the Aggregation Port's System Identifier. When the Aggregator does not support a DRNI Gateway, the operational value is always the same as the administrative value.
Data Type: Integer.
Value: 1 to 65535.

Actor_Port_Number
The Port Number used by LACP to uniquely identify the Aggregation Port within the Aggregation System. Port Number 0 is not assigned to any Aggregation Port. When this value is changed by management, the value of Selected variable is set to UNSELECTED, Actor_Oper_Port_State.Synchronization is set to FALSE, and Partner_Oper_Port_State.Synchronization is set to FALSE.
Data Type: Integer.

Value: 1 to 65535.
Administratively assigned.
Alias of aAggPortActorPort (7.3.2.1.14).

NOTE 2—The Actor_Port_Interface_ID and the Actor_Port_Number can be, but are not necessarily, the same value.

Actor_Port_Priority
The Port Priority value assigned to the Aggregation Port and used to converge dynamic Key changes. When this value is changed by management, the value of Selected variable is set to UNSELECTED, Actor_Oper_Port_State.Synchronization is set to FALSE, and Partner_Oper_Port_State.Synchronization is set to FALSE.
Data Type: Integer.
Value: 0 to 65535.
Administratively assigned.
Alias of aAggPortActorPortPriority (7.3.2.1.15).

Actor_Port_Aggregator_Identifier
The Aggregator_Interface_ID of the Aggregator most recently selected by this Aggregation Port. It is initialized to zero and updated whenever the selected variable changes from UNSELECTED or STANDBY to SELECTED. This value is reported in the aAggPortSelectedAggID (7.3.2.1.12) managed object when the selected variable is SELECTED and in the aAggPortAttachedAggID (7.3.2.1.13) managed object when the port_attached variable is TRUE.
Data Type: ifIndex.

Actor_Admin_Port_Key
The administrative value of Key administratively assigned to this Aggregation Port. When this value is changed by management, the value of Selected variable is set to UNSELECTED, Actor_Oper_Port_State.Synchronization is set to FALSE, and Partner_Oper_Port_State.Synchronization is set to FALSE.
Data Type: Integer.
Value: 1 to 65535.
Alias of aAggPortActorAdminKey (7.3.2.1.4).

Actor_Oper_Port_Key
The operational value of Key assigned to this Aggregation Port by the Actor.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPortActorOperKey (7.3.2.1.5)

Actor_Admin_Port_State
The administrative values of the Actor's state parameters. This consists of the following set of variables, as described in 6.4.2.3:
LACP_Activity
Short_Timeout
Aggregation
Synchronization
Collecting
Distributing
Defaulted
Expired
When management changes the LACP_Activity value in this variable, the new value is immediately copied to Actor_Oper_Port_State.LACP_Activity.
Data Type: 1 octet.
Alias of aAggPortActorAdminState (7.3.2.1.20).

Actor_Oper_Port_State
The operational values of the Actor's state parameters. This consists of the following set of variables, as described in 6.4.2.3:
LACP_Activity
Short_Timeout
Aggregation
Synchronization
Collecting
Distributing
Defaulted
Expired
Data Type: 1 octet.
Alias of aAggPortActorOperState (7.3.2.1.21).

Partner_Admin_System
Default value for the MAC address component of the System Identifier of the Partner, administratively assigned for use when the Partner's information is unknown or expired. When this variable is changed by management while Actor_Oper_Port_State.Defaulted is TRUE, the new value of Partner_Admin_System is copied to Partner_Oper_System, and Selected is set to UNSELECTED.
Data Type: 6 octets.
Alias of aAggPortPartnerAdminSystemID (7.3.2.1.8).

Partner_Oper_System
The operational value of the MAC address component of the System Identifier of the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU or to the value of Partner_Admin_System.
Data Type: 6 octets.
Alias of aAggPortPartnerOperSystemID (7.3.2.1.9).

Partner_Admin_System_Priority
Default value for the System Priority component of the System Identifier of the Partner, administratively assigned for use when the Partner's information is unknown or expired. When this variable is changed by management while Actor_Oper_Port_State.Defaulted is TRUE, the new value of Partner_Admin_System_Priority is copied to Partner_Oper_System_Priority, and Selected is set to UNSELECTED.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPortPartnerAdminSystemPriority (7.3.2.1.6).

Partner_Oper_System_Priority
The operational value of the System Priority of the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU or to the value of Partner_Admin_System_Priority.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPortPartnerOperSystemPriority (7.3.2.1.7).

Partner_Admin_Key
Default value for the Partner's Key, administratively assigned for use when the Partner's information is unknown or expired. When this variable is changed by management while Actor_Oper_Port_State.Defaulted is TRUE, the new value of Partner_Admin_Key is copied to Partner_Oper_Key and Selected is set to UNSELECTED.
Data Type: Integer.
Value: 1 to 65535.
Alias of aAggPortPartnerAdminKey (7.3.2.1.10).

Partner_Oper_Key
: The operational value of the Key value assigned to this link by the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU or to the value of Partner_Admin_Key.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPortPartnerOperKey (7.3.2.1.11).

Partner_Admin_Port_Number
: Default value for the Port Number component of the Partner's Port Identifier, administratively assigned for use when the Partner's information is unknown or expired. When this variable is changed by management while Actor_Oper_Port_State.Defaulted is TRUE, the new value of Partner_Admin_Port_Number is copied to Partner_Oper_Port_Number, and Selected is set to UNSELECTED.
Data Type: Integer.
Value: 1 to 65535.
Alias of aAggPortPartnerAdminPort (7.3.2.1.16).

Partner_Oper_Port_Number
: The operational value of the Port Number assigned to this link by the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU or to the value of Partner_Admin_Port_Number.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPortPartnerOperPort (7.3.2.1.17).

Partner_Admin_Port_Priority
: Default value for the Port Priority component of the Partner's Port Identifier, administratively assigned for use when the Partner's information is unknown or expired. When this variable is changed by management while Actor_Oper_Port_State.Defaulted is TRUE, the new value of Partner_Admin_Port_Priority is copied to Partner_Oper_Port_Priority, and Selected is set to UNSELECTED.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPortPartnerAdminPortPriority (7.3.2.1.18).

Partner_Oper_Port_Priority
: The operational value of the priority value assigned to this link by the Partner and used to converge dynamic Key changes. The Actor sets this variable either to the value received from the Partner in an LACPDU or to the value of Partner_Admin_Port_Priority.
Data Type: Integer.
Value: 0 to 65535.
Alias of aAggPortPartnerOperPortPriority (7.3.2.1.19).

Partner_Admin_Port_State
: Default value for the Partner's state parameters, administratively assigned for use when the Partner's information is unknown or expired. The value consists of the following set of variables, as described in 6.4.2.3:
    LACP_Activity
    Short_Timeout
    Aggregation
    Synchronization
    Collecting
    Distributing
    Defaulted
    Expired
Setting Partner_Admin_Port_State.Aggregation to FALSE (encoded as a 0) ensures that this will always be a Solitary link when the Partner's information is unknown. Setting Partner_Admin_Port_State.Synchronization to FALSE (encoded as a 0) prevents this port from

becoming active when the Partner's information is unknown. When Partner_Admin_Port_State.Synchronization or Partner_Admin_Port_State.Aggregation is changed by management while Actor_Oper_Port_State.Defaulted is TRUE, the new value of Partner_Admin_Port_State is copied to Partner_Oper_Port_State, and Selected is set to UNSELECTED.
Data Type: 1 octet.
Alias of aAggPortPartnerAdminState (7.3.2.1.22).

Partner_Oper_Port_State
The operational value of the Actor's view of the current values of the Partner's state parameters. The Actor sets this variable either to the value received from the Partner in an LACPDU or to the value of Partner_Admin_Port_State. The value consists of the following set of variables, as described in 6.4.2.3:
LACP_Activity
Short_Timeout
Aggregation
Synchronization
Collecting
Distributing
Defaulted
Expired
Data Type: 1 octet.
Alias of aAggPortPartnerOperState (7.3.2.1.23).

Partner_LACP_Version
The Version number of LACP as reported by the Partner. The Actor sets this variable either to the value in the most recently received LACPDU or to the default value 1.
Data Type: Integer.
Value: 0 to 255.
Alias of aAggPortPartnerLacpVersion (7.3.2.1.34).

Wtr_Time
The amount of time (in seconds) an Aggregation Port currently attached to an Aggregator waits after Port_Operational becomes FALSE and then TRUE again before once more becoming active in the LAG.
Data Type: Integer.
Value: 0 to 32767.
Administratively assigned.
Alias of aAggPortWtrTime (7.3.2.1.30).

Wtr_Revertive
An administrative control determining whether the Aggregation Port is revertive (i.e., can become active in a LAG as soon as the wtrWhile timer has expired regardless of the state of other links in the LAG) or non-revertive (i.e., will not become active on an Aggregator until there are no other operational links that are, or could become, active on the Aggregator).
Data Type: Boolean.
Alias of aAggPortWtrRevertive (7.3.2.1.31).

Wtr_Waiting
An indication from the Mux machine that the link is being held in the MUX:ATTACHED_WTR state with Collecting and Distributing disabled. If Wtr_Revertive is FALSE, the Aggregation Port will no longer be used unless the Selection Logic clears Wtr_Waiting, or Selection Logic changes Selected to UNSELECTED or STANDBY to detach the Aggregation Port from the current Aggregator and subsequently selects the same or a different Aggregator.
Data Type: Boolean.
Alias of aAggPortWtrWaiting (7.3.2.1.32).

### 6.4.7 Variables used for managing the operation of the state machines

BEGIN
This variable indicates the initialization (or reinitialization) of the LACP entity. It is set to TRUE when the System is initialized and is set to FALSE when initialization has completed.
Data Type: Boolean.

The following variables have one instance per Aggregation Port:

Port_Operational
An alias for the MAC_Operational status parameter of the Aggregation Port ISS.
Data Type: Boolean.
TRUE if the underlying service is operable.
FALSE otherwise.

NOTE 1—The means by which the value of the Port_Operational variable is generated by the underlying service [e.g., a MAC or Maintenance End Point (MEP) (Clause 18 of IEEE Std 802.1Q-2018)] is implementation dependent.

NTT
Need To Transmit flag.
Data Type: Boolean.
TRUE indicates that there is new protocol information to be transmitted on the link or that the Partner needs to be reminded of the old information.
FALSE otherwise.

LACP_Timeout
The timeout value for the LACP Receive machine timer, set to either Short_Timeout_Time or Long_Timeout_Time by the LACP Receive machine according to the Actor_Oper_Port_State.Short_Timeout value (which is determined by the Actor_Admin_Port_State.Short_Timeout value).

Ready
The Selection Logic asserts Ready TRUE to indicate to the MUX machine that the Aggregation Port is to be attached to the selected Aggregator. Once asserted, Ready remains TRUE until the MUX machine enters the DETACHED state as a result of the port being UNSELECTED.
Data Type: Boolean.

Selected
A value of SELECTED indicates that the Selection Logic has selected an appropriate Aggregator. A value of UNSELECTED indicates that no Aggregator is currently selected. A value of STANDBY indicates that although the Selection Logic has selected an appropriate Aggregator, aggregation restrictions currently prevent the Aggregation Port from being enabled as part of the aggregation, and so the Aggregation Port is being held in a standby condition. This variable can be set to SELECTED or STANDBY only by the operation of the Aggregation Port's Selection Logic. It can be set to UNSELECTED by the operation of the Aggregation Port's LACP Receive machine or by the operation of the Selection Logic associated with another Aggregation Port.

NOTE 2—Setting Selected UNSELECTED in the Selection Logic associated with another Aggregation Port occurs if the Selection Logic determines that the other Aggregation Port has a stronger claim to attach to this Aggregation Port's current Aggregator.

Value: SELECTED, UNSELECTED, or STANDBY.

port_attached
: This variable is set to TRUE when the process of attaching an Aggregation Port to an Aggregator, initiated by the attachMuxToAggregator function, has completed. This variable is set to FALSE when the process of detaching an Aggregation Port from an Aggregator is initiated by the detachMuxFromAggregator function.
Data Type: Boolean

NOTE 3—This definition of port_attached means that the variable is TRUE only when the Aggregation Port is attached to the Aggregator and is FALSE when not attached or in transition (i.e., during the process of attaching or detaching). This introduces the possibility that the Mux state machine for an Aggregation Port could issue a request to start the process of attaching to a new Aggregator before the process of detaching from an old Aggregator has completed. In that case the request to attach to the new Aggregator is honored, but delayed until the process of detaching from the old Aggregator has completed.

txOpportunity
: A Boolean signal from the System that is TRUE when LACPDU transmission is permitted and FALSE when transmission of LACPDUs is inhibited due to transmit resource contention and/or protocol rate limiting.

NOTE 4—Controlling txOpportunity is a mechanism for a System to enforce the Slow Protocols rate limit of no more than ten PDUs transmitted per second by any protocol using the Slow Protocols Address and EtherType (see IEEE Std 802.3-2018 Annex 57A).

### 6.4.8 Functions

In the following function definitions, values obtained from fields in a received LACPDU are shown in italics to avoid confusion with local variables having a similar name.

recordPDU
: In the process of executing the recordPDU function, an LACP Receive machine compliant to this standard shall not validate the *Reserved* fields of any TLV or the *TLV_Type* or *TLV_Length* fields of the first three TLVs (*Actor Information TLV*, *Partner Information TLV*, and *Collector Information TLV*). The same actions are taken regardless of the values received in these fields. If the *Version Number* in the received LACPDU is 1, an LACP Receive machine shall not validate the *TLV_Type* or *TLV_Length* fields of the *Terminator TLV*. If the *Version Number* in the received LACPDU is greater than 1, an LACP Receive machine shall validate the *TLV_Type* and *TLV_Length* fields of all TLVs following the first three TLVs. A TLV with reserved or deprecated *TLV_Type*, or with a *TLV_Length* value that does not match the expected length of the TLV, is ignored; and processing continues with the next TLV (using the *TLV_Length* in the LACPDU to find the next TLV). Processing of the TLVs terminates when a *Terminator TLV* or the end of the LACPDU is encountered.

This function records the parameter values for the Actor carried in a received LACPDU (*Version_Number*, *Actor_Port_Number*, *Actor_Port_Priority*, *Actor_System*, *Actor_System_Priority*, *Actor_Key*, and *Actor_State* variables) as the current Partner operational parameter values (Partner_LACP_Version, Partner_Oper_Port_Number, Partner_Oper_Port_Priority, Partner_Oper_System, Partner_Oper_System_Priority, Partner_Oper_Key, and Partner_Oper_Port_State variables with the exception of Synchronization) and sets both Actor_Oper_Port_State.Expired and Actor_Oper_Port_State.Defaulted to FALSE.

This function also updates the value of the Partner_Oper_Port_State.Synchronization using the parameter values carried in received LACPDUs. Parameter values for the Partner carried in the received PDU (*Partner_Port*, *Partner_Port_Priority*, *Partner_System*, *Partner_System_Priority*, *Partner_Key*, and *Partner_State.Aggregation*) are compared to the corresponding operational parameter values for the Actor (Actor_Port_Number, Actor_Port_Priority, Actor_Admin_Port_System, Actor_Admin_Port_System_Priority,

Actor_Oper_Port_Key, and Actor_Oper_Port_State.Aggregation). Partner_Oper_Port_State.Synchronization is set to TRUE if all of these parameters match, *Actor_State.Synchronization* in the received LACPDU is TRUE, and LACP will actively maintain the link in the aggregation.

Partner_Oper_Port_State.Synchronization is also set to TRUE if the value of *Actor_State.Aggregation* in the received LACPDU is FALSE (i.e., indicates a Solitary link), *Actor_State.Synchronization* in the received LACPDU is TRUE, and LACP will actively maintain the link.

Otherwise, Partner_Oper_Port_State.Synchronization is set to FALSE.

LACP is considered to be actively maintaining the link if either the received PDU's *Actor_State.LACP_Activity* variable is TRUE or both the Actor's Actor_Oper_Port_State.LACP_Activity and the LACPDU's *Partner_State.LACP_Activity* variables are TRUE.

The LACP_Timeout value is set to Short_Timeout_Time if Actor_Oper_Port_State.Short_Timeout is TRUE and set to Long_Timeout_Time otherwise.

If Conversation-Sensitive Collection and Distribution is supported, this function executes the recordPortCSCD function.

recordDefault

This function records the default parameter values for the Partner carried in the Partner Admin parameters (Partner_Admin_Port_Number, Partner_Admin_Port_Priority, Partner_Admin_System, Partner_Admin_System_Priority, Partner_Admin_Key, and Partner_Admin_Port_State) as the current Partner operational parameter values (Partner_Oper_Port_Number, Partner_Oper_Port_Priority, Partner_Oper_System, Partner_Oper_System_Priority, Partner_Oper_Key, and Partner_Oper_Port_State). It also sets the Partner_LACP_Version to 1 and sets both Partner_Oper_Port_State.Synchronization and Partner_Oper_Port_State.Collecting to the value of Partner_Admin_Port_State.Synchronization.

This function updates the Actor_Oper_Port_State by setting Expired to FALSE, Defaulted to TRUE, Short_Timeout to the value of Actor_Admin_Port_State.Short_Timeout, LACP_Activity to the value of Actor_Admin_Port_State.LACP_Activity, and Aggregation to TRUE if both Actor_Admin_Port_State.Aggregation and the ISS operPointToPointMAC variable are TRUE.

If Conversation-Sensitive Collection and Distribution is supported, this function executes the recordDefaultPortCSCD function.

transmitLACPDU

This function generates an LACPDU, formatted as specified in 6.4.2, and issues it as a CtrlMuxN:M_UNITDATA.Request(LACPDU) service primitive to the LACP Parser/ Multiplexer (6.2.9).

updateSelected

This function updates the value of the Selected variable by using parameter values from a newly received LACPDU. The parameter values for the Actor carried in the received LACPDU (*Actor_Port*, *Actor_Port_Priority*, *Actor_System*, *Actor_System_Priority*, *Actor_Key*, and *Actor_State.Aggregation*) are compared with the corresponding operational parameter values for the Aggregation Port's Partner (Partner_Oper_Port_Number, Partner_Oper_Port_Priority,

Partner_Oper_System, Partner_Oper_System_Priority, Partner_Oper_Key, and Partner_Oper_Port_State.Aggregation). If one or more of the comparisons show that the value(s) received in the LACPDU differ from the current operational values, then Selected is set to UNSELECTED; otherwise, Selected remains unchanged.

updateNTT

This function updates the value of the NTT variable by using parameter values from a newly received LACPDU. The parameter values for the Partner carried in the received LACPDU (*Partner_Port*, *Partner_Port_Priority*, *Partner_System*, *Partner_System_Priority*, *Partner_Key*, *Partner_State.LACP_Activity*, *Partner_State.Short_Timeout*, *Partner_State.Synchronization*, and *Partner_State.Aggregation*) are compared with the corresponding operational parameter values for the Actor (Actor_Port_Number, Actor_Port_Priority, Actor_Oper_Port_System, Actor_Oper_Port_System_Priority, Actor_Oper_Port_Key, Actor_Oper_Port_State.LACP_Activity, Actor_Oper_Port_State.Short_Timeout, Actor_Oper_Port_State.Synchronization, and Actor_Oper_Port_State.Aggregation). If one or more of the comparisons show that the value(s) received in the LACPDU differ from the current operational values, then NTT is set to TRUE; otherwise, NTT remains unchanged.

attachMuxToAggregator

This function initiates the process of attaching the Aggregation Port's LACP Parser/Multiplexer to the Aggregator Parser/Multiplexer of the selected Aggregator, in preparation for collecting and distributing frames. The port_attached variable is set to TRUE when the attachment process completes.

detachMuxFromAggregator

This function initiates the process of detaching the Aggregation Port's LACP Parser/Multiplexer from the Aggregator Parser/Multiplexer of the Aggregator to which the Aggregation Port is currently attached. The port_attached variable is set to FALSE when the detachment process is initiated.

enableCollecting

This function causes the Aggregator Parser of the Aggregator to which the Aggregation Port is attached to start collecting frames from the Aggregation Port and causes the Aggregator Multiplexer to stop distributing frames to the Aggregation Port. Then Actor_Oper_Port_State.Collecting is set to TRUE. If this is a change to Actor_Oper_Port_State.Collecting and Conversation-Sensitive Collection and Distribution is supported, this function also sets the selected Aggregator's changeDistAlg and changeLinkState flags to TRUE.

enableCollectingDistributing

This function causes the Aggregator Parser of the Aggregator to which the Aggregation Port is attached to start collecting frames from the Aggregation Port and causes the Aggregator Multiplexer to start distributing frames to the Aggregation Port. Then Actor_Oper_Port_State.Collecting is set to TRUE. If this is a change to Actor_Oper_Port_State.Collecting and Conversation-Sensitive Collection and Distribution is supported, this function also sets the selected Aggregator's changeDistAlg and changeLinkState flags to TRUE.

disableCollectingDistributing

This function causes the Aggregator Parser of the Aggregator to which the Aggregation Port is attached to stop collecting frames from the Aggregation Port and causes the Aggregator Multiplexer to stop distributing frames to the Aggregation Port. Then Actor_Oper_Port_State.Collecting is set to FALSE. If this is a change to

Actor_Oper_Port_State.Collecting and if Conversation-Sensitive Collection and Distribution is supported, this function also clears the Distribution_Conversation_Mask, the Collection_Conversation_Mask, and the Port_Oper_Conversation_Mask; copies the value of Admin_Link_Number to Link_Number; and sets the selected Aggregator's changeDistAlg and changeLinkState flags to TRUE.

### 6.4.9 Timers

LACP_currentWhile
: This timer is used by the LACP Receive machine to detect whether received protocol information has expired. If Actor_Oper_Port_State.Short_Timeout is TRUE, the timer is started with the value Short_Timeout_Time; otherwise, it is started with the value Long_Timeout_Time (see 6.4.4).

LACP_txWhen
: This timer is used by the LACP Transmit machine to generate periodic transmissions. It is started using the value Slow_Periodic_Time or Fast_Periodic_Time (see 6.4.4), as specified in the LACP Transmit machine.

waitWhile
: This timer can be used by the Selection Logic to provide hysteresis before performing an aggregation change so that all links that will join this LAG can do so at the same time. It is started using the value Aggregate_Wait_Time (see 6.4.4).

wtrWhile
: The wait-to-restore timer used by the Mux machine to prevent an Aggregation Port that is attached to an Aggregator from becoming active in the LAG until it has had Port_Operational continuously TRUE for a Wtr_Time.

### 6.4.10 Messages

rcvdLACPDU
: This message is generated by the LACP Parser/Multiplexer as a result of the reception of an LACPDU (formatted as defined in 6.4.2) while the operPointToPointMAC parameter of the Aggregation Port ISS is TRUE.

### 6.4.11 LACP Receive machine

The LACP Receive machine shall implement the function specified in Figure 6-14 with its associated parameters (6.4.4 through 6.4.10).

On receipt of an LACPDU, the state machine enters the CURRENT state. The update_Selected function sets the Selected variable to UNSELECTED if the Actor's view of the Partner's operational parameters is not up to date. The Selected variable is used by the Mux machine (6.4.13).

NOTE—The LACP Receive machine can set the Selected variable to UNSELECTED; however, setting this variable to SELECTED or STANDBY is the responsibility of the Selection Logic.

The update_NTT function is used to determine whether further protocol transmissions are required; NTT is set to TRUE if the Partner's view of the Actor's operational parameters is not up to date. The recordPDU function records the information contained in the LACPDU in the Partner operational variables, and the LACP_currentWhile timer is started. The value used to start the timer is either Short_Timeout_Time or Long_Timeout_Time, depending upon Actor_Oper_Port_State.Short_Timeout.

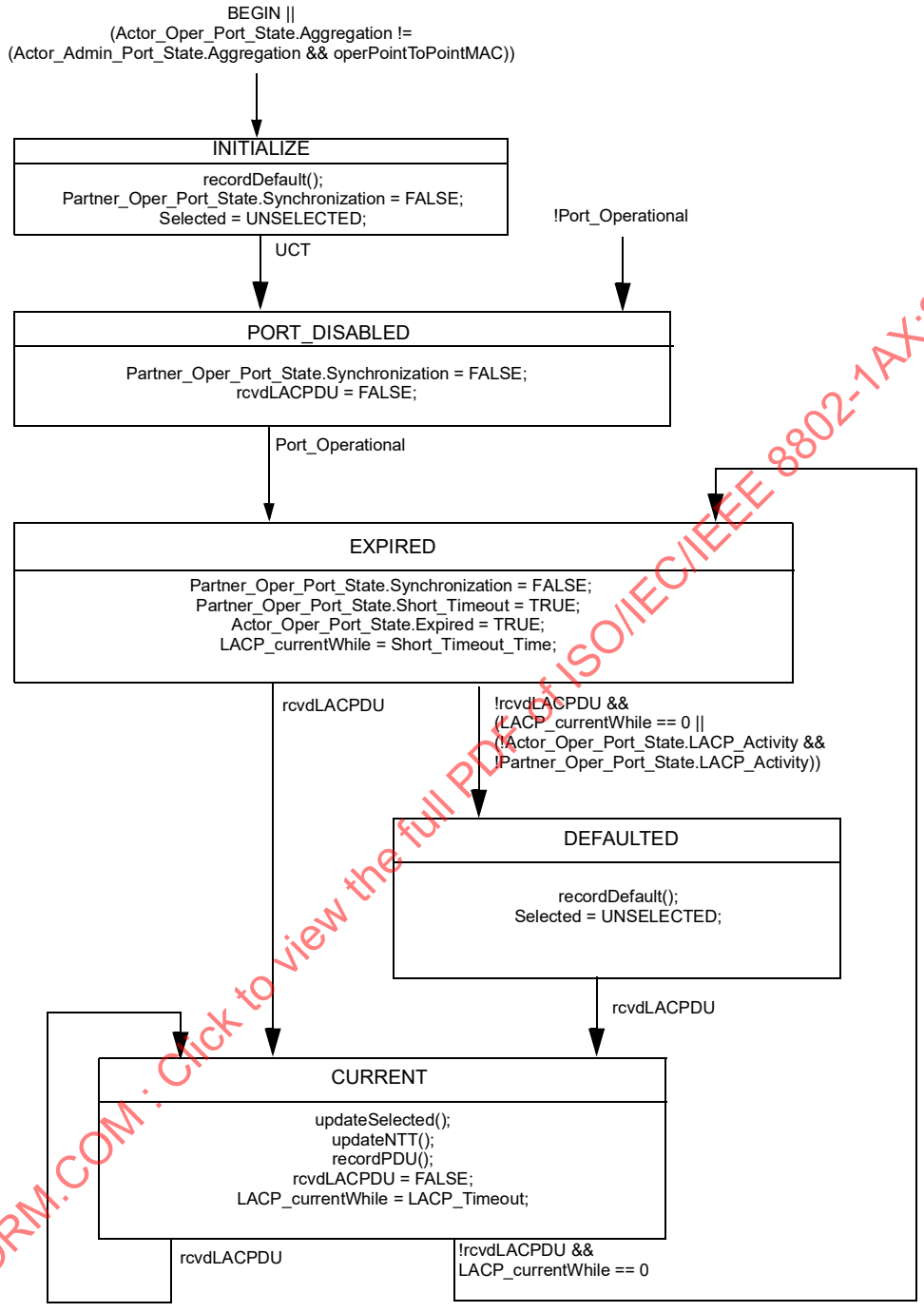





**Figure 6-14—LACP Receive state diagram**

If no LACPDU is received before the LACP_currentWhile timer expires, the state machine transits to the EXPIRED state. The Partner_Oper_Port_State.Synchronization variable is set to FALSE. The Partner_Oper_Port_State.Short_Timeout variable is set to TRUE so that the LACP Transmit machine will use fast periodic transmissions in an attempt to reestablish communication with the Partner. The LACP_currentWhile timer is started with a value of Short_Timeout_Time.

If no LACPDU is received before the LACP_currentWhile timer expires again, the state machine transits to the DEFAULTED state. The Selected variable is set to UNSELECTED. The recordDefault function overwrites the current operational parameters for the Partner with administratively configured values. This allows configuration of aggregations and Solitary links when no protocol Partner is present, while still permitting an active Partner to override default settings. Since all operational parameters are now set to locally administered values, there can be no disagreement about the LAG; therefore, the Partner_Oper_Port_State.Synchronization variable is set to TRUE.

If the Aggregation Port becomes inoperable and the BEGIN variable is not asserted, the state machine enters the PORT_DISABLED state. Partner_Oper_Port_State.Synchronization is set to FALSE. This state allows the current Selection state to remain undisturbed, so that, in the event that the Aggregation Port is still connected to the same Partner and Partner Aggregation Port when it becomes operable again, there will be no disturbance caused to higher layers by unnecessary reconfiguration.

The INITIALIZE state is entered following a BEGIN event. The INITIALIZE state is also entered if the operational state indicates the link can be aggregated (Actor_Oper_Port_State.Aggregation is TRUE) but the administrative state indicates that it is to be treated as a Solitary link (Actor_Admin_Port_State.Aggregation is FALSE) or the link is not a point-to-point link. Conversely, the INITIALIZE state is also entered if the operational state indicates that the link is a Solitary link, but the administrative state indicates that it is Aggregateable and the link is a point-to-point link. This state causes the administrative values of the Partner parameters to be used as the current operational values and sets Selected to UNSELECTED. These actions force the Mux machine to detach the Aggregation Port from its current Aggregator. The state machine immediately transitions to the PORT_DISABLED state.

### 6.4.12 Selection Logic

The Selection Logic selects a compatible Aggregator for an Aggregation Port by using the Aggregation Port's LAG ID. The Selection Logic selects an Aggregator for an Aggregation Port when the port is not currently attached to an Aggregator and the value of Ready is FALSE for that Aggregation Port. The Selection Logic can determine that the link is to be maintained as a standby link if there are constraints on the simultaneous attachment of Aggregation Ports that have selected the same Aggregator. The Selection Logic can also impose a delay before committing an Aggregation Port to a selected Aggregator to give the selection criteria time to stabilize or to give other Aggregation Ports the opportunity to attach to the Aggregator at the same time.

NOTE 1—There will never be more than one Aggregator with the same LAG ID, but there can be none. Normally, the latter will be a temporary state, caused by the fact that it takes a finite time for Aggregation Ports to be moved to the correct Aggregators during reconfiguration.

The Selection Logic commits an Aggregation Port to a selected Aggregator by setting the value of Selected to SELECTED and Ready to TRUE. This authorizes the Mux machine to attach the Aggregation Port to the Selected Aggregator. Once attached, the Aggregation Port remains attached to the Aggregator as long as the value of Selected remains SELECTED. Changing the value of Selected from SELECTED causes the Mux machine to detach the Aggregation Port from the Aggregator and set the value of Ready to FALSE. At this point the Selection Logic can select a new Aggregator for the Aggregation Port.

The value of the Selected variable can be changed by the following:

a) *The LACP Receive machine*. The LACP Receive machine can set Selected to UNSELECTED at any time if there is a change to the partner components of the LAG ID (Partner System Identifier, Partner Key, Partner_State.Aggregation). This can be the result of receiving an LACPDU with new partner values or transitioning to a state that sets the partner values to the administrative default values.

b) *Management action*. Selected can be set to UNSELECTED at any time in response to any management operation that changes the values of the actor components of the LAG ID (Actor System Identifier, Actor Key, or Actor_State.Aggregation) or that disables the Aggregator that has been selected for an Aggregation Port.

c) *The Selection Logic*. When selecting an Aggregator for an Aggregation Port, the Selection Logic can change the value of Selected between UNSELECTED, SELECTED, and STANDBY in response to changes in the selection criteria. When selecting an Aggregator for another Aggregation Port, the Selection Logic can change the value of Selected to UNSELECTED or STANDBY for an Aggregation Port that has already selected the Aggregator if the Selection Logic determines that the other Aggregation Port has a better claim to the Aggregator.

NOTE 2—The STANDBY value for the Selected variable is provided for the convenience of the Selection Logic to identify Aggregation Ports for which the Selection Logic has conditionally selected an Aggregator, but that currently cannot be attached to the Aggregator due to system constraints. The value of STANDBY has no significance except to the Selection Logic itself. For the Mux machine (the only other consumer of the Selected variable), STANDBY is identical to UNSELECTED.

NOTE 3—An Aggregation Port is always detached from its prior Aggregator when the LAG ID changes, even if the same Aggregator is selected later; to do otherwise would be to risk misdelivery of frames. Selection of a new Aggregator cannot take place until the Aggregation Port is detached from any prior Aggregator. Other Aggregators can become free while the Aggregation Port is detaching, and other Aggregation Ports can attach to some of the available Aggregators during this time interval.

The operation of the Selection Logic is separated into the following two subclauses:

— The requirements for the correct operation of the Selection Logic are defined in 6.4.12.1.
— The recommended default operation of the Selection Logic is described in 6.4.12.2.

This separation reflects the fact that a wide choice of selection rules is possible within the proper operation of the protocol. An implementation that claims conformance to this standard may support selection rules other than the recommended default; however, any such rules shall meet the requirements stated in 6.4.12.1.

### 6.4.12.1 Selection Logic—Requirements

The Selection Logic selects an appropriate Aggregator for an Aggregation Port and sets Ready for that Aggregation Port to TRUE to cause the Mux machine to attach it to that Aggregator. The following are required for correct operation of the selection and attachment logic:

a) The implementation shall support at least one Aggregator per System.
b) Each Aggregation Port shall be assigned an operational Key (6.3.5). Aggregation Ports that can aggregate together are assigned the same operational Key as the other Aggregation Ports with which they can aggregate. Aggregation Ports that cannot aggregate with any other Aggregation Port are allocated unique operational Keys.
c) Each Aggregator shall be assigned an operational Key.
d) Each Aggregator shall be assigned an identifier that distinguishes it among the set of Aggregators in the System.
e) An Aggregation Port shall select only an Aggregator that has the same System Identifier and operational Key assignment as its own System Identifier and operational Key.
f) Subject to the exception stated in item g), Aggregation Ports that are members of the same LAG (i.e., two or more Aggregation Ports that have the same Actor System Identifier, Actor Key, Partner System Identifier, and Partner Key, and that are not required to be Solitary) shall select the same Aggregator.

g) An Aggregation Port (P) whose Partner System Identifier is the same as its Actor System Identifier shall not select the same Aggregator as a different Aggregation Port (Q) that has an Actor Port Identifier that is the same as P's Partner Port Identifier.

NOTE 1—This exception avoids attaching the two ends of a single link to the same Aggregator, with the undesirable consequence of one Aggregation Port receiving a frame transmitted through another Aggregation Port when both Aggregation Ports are attached to the same Aggregator. It permits the use of both Solitary and aggregated loopback links, where a frame transmitted by a system on one Aggregator is received on another Aggregator.

h) The selection of an Aggregator by an Aggregation Port shall not result in any Aggregation Port that is required to be Solitary having selected the same Aggregator as another Aggregation Port.

i) An Aggregation Port shall not select an Aggregator that has been disabled by management (Aggregator_Enabled set to FALSE). If any Aggregation Ports have selected an Aggregator at the point when management sets Aggregator_Enabled to FALSE, the Selected variable of those Aggregation Ports is set to UNSELECTED.

j) If the preceding conditions (a through i) result in a given Aggregation Port being unable to select an Aggregator, then that Aggregation Port shall not be attached to any Aggregator.

k) If there are further constraints on the attachment of Aggregation Ports that have selected an Aggregator, those Aggregation Ports can be selected as standby in accordance with the rules specified in 6.7.1. Selection or deselection of that Aggregator can cause the Selection Logic to reevaluate the Aggregation Ports to be selected as standby.

l) The Selection Logic operates upon the operational information recorded by the Receive state machine, along with knowledge of the Actor's own operational configuration and state. The Selection Logic uses the LAG ID for the Aggregation Port, determined from these operational parameters, to locate the correct Aggregator to which to attach the Aggregation Port.

m) The Selection Logic selects an Aggregator for an Aggregation Port when that Aggregation Port is not currently attached (port_attached is FALSE), and has not been authorized to attach (Ready is FALSE), to an Aggregator. While selecting an Aggregator, the Selection Logic can change the Selected variable to any value indicating that it has not selected (UNSELECTED), has selected (SELECTED), or has conditionally selected (STANDBY) a suitable Aggregator.

NOTE 2—The Selection Logic can take a significant time to complete its determination of the correct Aggregator, as a suitable Aggregator might not be immediately available, due to configuration restrictions or the time taken to reallocate Aggregation Ports to other Aggregators.

n) The Selection Logic can use the waitWhile timer to delay the attachment of an Aggregation Port to an Aggregator so that other Aggregation Ports that could potentially attach to the same Aggregator can attach at the same time.

o) Once the correct Aggregator has been determined, the variable Selected shall be set to SELECTED, and the variable Ready shall be set to TRUE. The Selection Logic shall not change the selected Aggregator, or change Selected from UNSELECTED or STANDBY to SELECTED, while Ready is TRUE.

NOTE 3—When Selected is SELECTED and Ready is TRUE, the Mux machine starts the process of attaching the Aggregation Port to the selected Aggregator.

p) Where the selection of a new Aggregator for an Aggregation Port, as a result of changes to the selection parameters, results in other Aggregation Ports in the System being required to reselect their Aggregators in turn, this is achieved by setting Selected to UNSELECTED for those other Aggregation Ports that are required to reselect their Aggregators.

q) When the Selection Logic changes Selected from SELECTED to UNSELECTED for an Aggregation Port with Wtr_Waiting TRUE and Wtr_Revertive FALSE, that Aggregation Port is considered to be non-revertive. The Selection Logic shall not select an Aggregator for a non-revertive Aggregation Port if Wtr_Revertive remains FALSE and there are any other Aggregation Ports with a claim to that Aggregator that are operational with Wtr_Waiting FALSE and are not considered to be non-revertive.

r) When a DRNI has been formed by a pair of DRNI Systems (9.5.3.2), the DRNI System with the numerically higher Home System Identifier (9.5.3.1) shall not select an Aggregator for an Aggregation Port connected to an LACP Partner if the other DRNI System has one or more active links to a different LACP Partner.

#### 6.4.12.2 Selection Logic—Recommended default operation

The recommended default behavior balances the goal of making the current configuration independent of past history with that of minimizing the disruption experienced by Aggregator Clients as a consequence of changes to Port_Operational for Aggregation Ports. It also has the characteristic that no additional MAC addresses are needed for Aggregators over and above those already assigned to Aggregation Ports.

NOTE—This standard does not specify any alternative selection rules beyond the recommended set. A wide variety of selection rules are possible within the scope of the requirements stated in 6.4.12.1. In particular, it is possible within these requirements to support implementations that provide fewer Aggregators than Aggregation Ports, as well as implementations designed to minimize configuration changes at the expense of less deterministic behavior.

In the default configuration, each Aggregation Port has an Aggregator associated with it (i.e., the number of Aggregators in the System equals the number of Aggregation Ports supported), and each Aggregation Port and each Aggregator are assigned the same operational Key value. This configuration can be altered to assign either of the following:

— Different Key values to different sets of Aggregation Ports when it is not possible or not desirable to place Aggregation Ports from the different sets into the same Link Aggregation Group, or
— A given Key value to fewer Aggregators than Aggregation Ports (e.g., to support dual-homed configurations described in 6.7.5).

An Aggregator is selected from the set of Aggregators with the same operational Key value as the Aggregation Port. When multiple Aggregation Ports have the same selection parameters, they will select the same Aggregator. These selection parameters are as follows:

a) Actor's System Identifier
b) Actor's operational Key
c) Partner's System Identifier
d) Partner's operational Key

Whenever possible, the Aggregator selected is the Aggregator associated with the lowest numbered Aggregation Port in the set with the same selection parameters. Note that this lowest Numbered Aggregation Port might not be in a state that allows data transfer across the link; however, it has selected the Aggregator in question. This is illustrated in Figure 6-15. If it is not possible to select the Aggregator associated with the lowest numbered Aggregation Port (e.g., if the lowest numbered Aggregation Port and its associated Aggregator have different operational Key values), then the Aggregator selected can be any Aggregator with a matching Key value that has not been selected by any Aggregation Ports with Port_Operational set to True. If such an Aggregator cannot be found, then the Aggregation Port does not select an Aggregator until the situation changes and an appropriate Aggregator can be found.

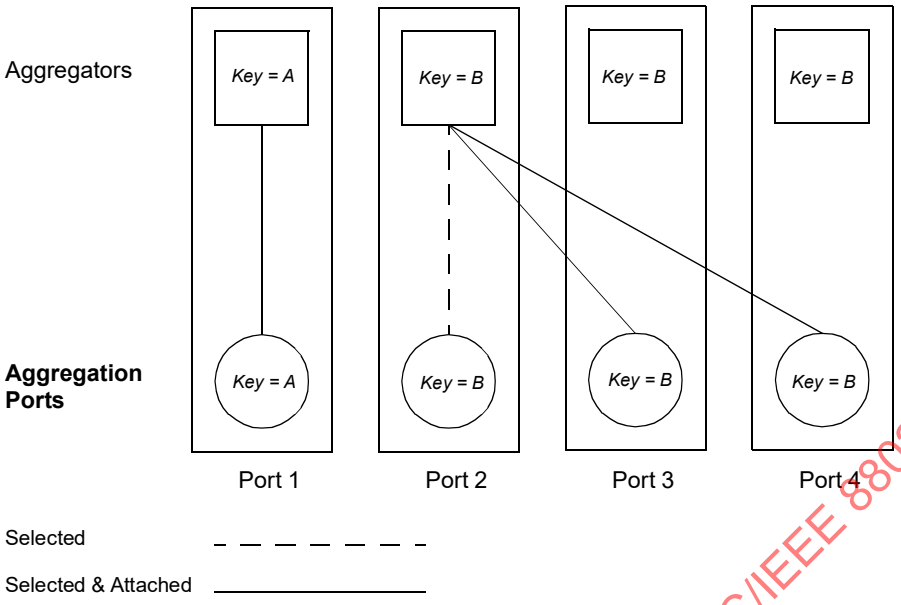



**Figure 6-15—Selection of Aggregators**

### 6.4.13 Mux machine

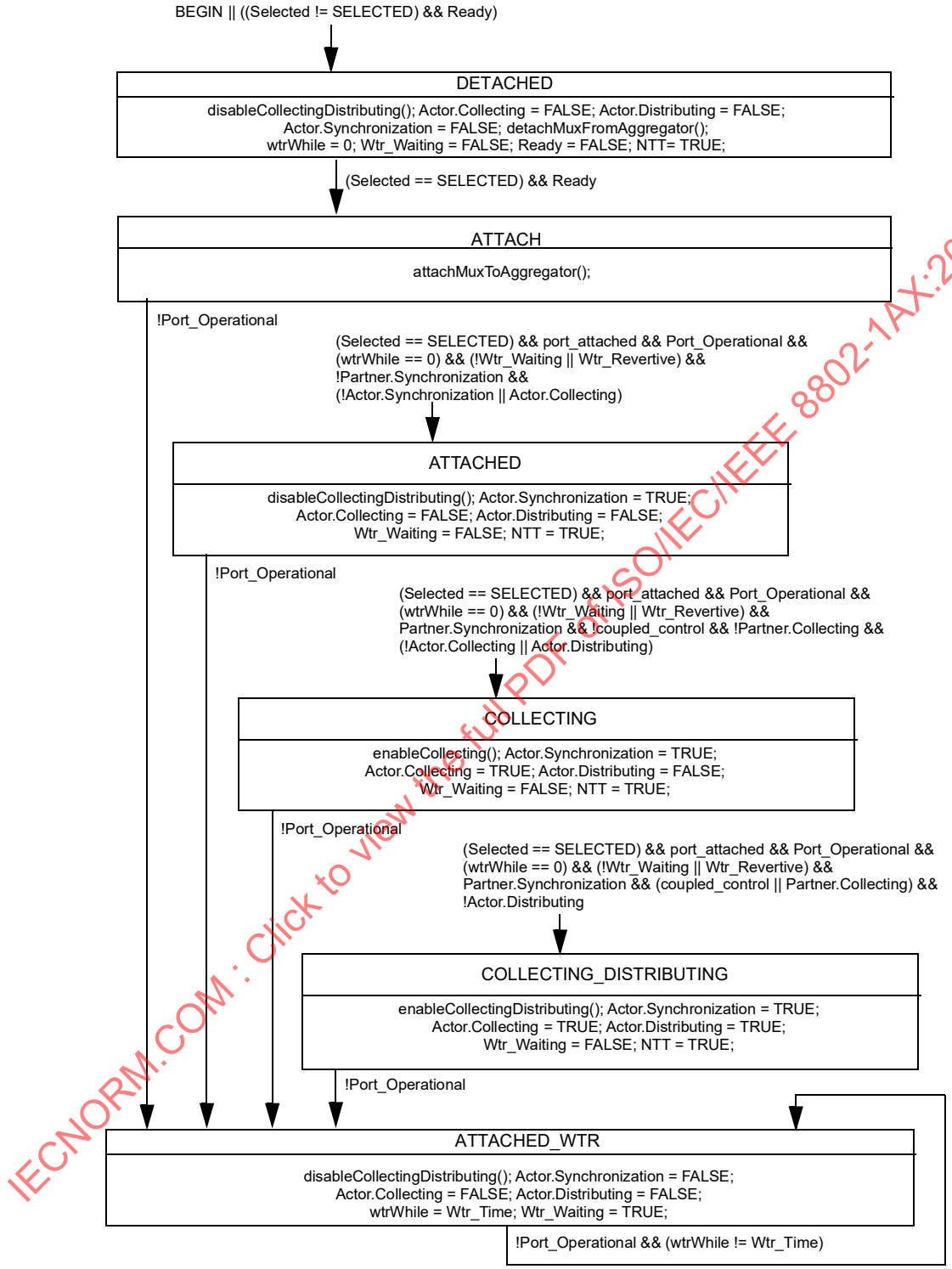
The Mux machine shall implement the function specified in Figure 6-16 with its associated parameters (6.4.4 through 6.4.10).

The state machine conventions in IEEE Std 802.1Q require that all in-state actions are performed in sequence and are atomic, completing before the evaluation of any exit conditions and before the execution of procedures or evaluation of exit conditions for any other state or state machine.

The Mux machine state diagram (Figure 6-16) is suitable for implementations that can control frame collection from an Aggregation Port, and frame distribution to an Aggregation Port, independently (coupled_control is FALSE), as well as implementations where collecting and distributing cannot be controlled independently (coupled_control is TRUE). Implementations should support a coupled_control value of FALSE.

The primary input to the Mux machine is the Selected variable. The LACP Receive machine can set Selected to UNSELECTED at any time if there is a change to any of the components of the LAG ID determined by the Aggregation Port (Actor_Oper_Port_System_Priority, Actor_Oper_Port_System, Actor_Oper_Port_Key, Actor_Port_Priority, Actor_Port_Number, Actor_Oper_Port_State.Aggregation, Partner_Oper_System_Priority, Partner_Oper_System, Partner_Oper_Key, Partner_Oper_Port_Priority, Partner_Oper_Port_Number, Partner_Oper_Port_State.Aggregation); otherwise, the value of Selected is controlled by the Selection Logic. The Selection Logic can also set Selected to UNSELECTED at any time if it determines that another Aggregation Port has a higher claim to the Aggregator currently selected for this Aggregation Port.

Whenever Selected has a value other than SELECTED (i.e., UNSELECTED or STANDBY), the Mux machine enters the DETACHED state. The collection of frames from the Aggregation Port and the distribution of frames to the Aggregation Port are disabled. The Actor_Oper_Port_State.Distributing, Actor_Oper_Port_State.Collecting, and Actor_Oper_Port_State.Synchronization variables are set to FALSE. The process of detaching the Aggregation Port from the Aggregator is started, port_attached is set to FALSE, and Ready is set to FALSE. The state machine remains in the DETACHED state until the Selection Logic is able to select an appropriate Aggregator.






The following abbreviations are used in this diagram:
“Actor” = Actor_Oper_Port_State
“Partner” = Partner_Oper_Port_State

**Figure 6-16—Mux state diagram**

The Selection Logic selects an Aggregator when the Mux machine is in the DETACHED state, with port_attached and Ready both FALSE. During the selection process, the Selection Logic can set Selected to any value as selection parameters change over time. Upon finding an appropriate Aggregator, the Selection Logic sets Selected to SELECTED, but can force a delay to allow for the possibility that other Aggregation Ports can also be selecting the same Aggregator and to synchronize the attachment of those Aggregation Ports. When the Selection Logic has set Selected to SELECTED and is prepared to commit the Aggregation Port to the selected Aggregator, it sets the Ready variable to TRUE. This allows the Mux machine to enter the ATTACH state.

On entry to the ATTACH state, the Mux machine initiates the process of attaching the Aggregation Port to the selected Aggregator. Once the attachment process has completed, the port_attached variable is set to TRUE.

If Port_Operational is FALSE when the port_attached variable is set to TRUE, or if Port_Operational becomes FALSE at any time when the port_attached variable is TRUE, the state machine enters the ATTACHED_WTR state. The wait-to-restore timer (wtrWhile) is started with the administered wait-to-restore interval (Wtr_Time), and the Wtr_Waiting flag is set to TRUE. The wait-to-restore timer provides hysteresis on the Port_Operational signal that protects the LAG from repeated redistribution of conversations among the aggregation links when a link is unstable. When the state machine is configured for non-revertive operation (Wtr_Revertive is FALSE), conversations that have been redistributed from a link in a LAG that became non-operational are not redistributed back to the link when it becomes operational unless the Selection Logic determines that there are no other links in the LAG actively collecting and distributing frames. While Port_Operational is FALSE, the state machine re-enters the ATTACHED_WTR state and restarts the wtrWhile timer whenever the wtrWhile timer is decremented (once per second). The state machine remains in the ATTACHED_WTR state until Port_Operational has been continuously TRUE for a wait-to-restore interval and, if configured for non-revertive operation, until the Selection Logic has set the Wtr_Waiting flag to FALSE.

When the Aggregation Port is attached to the Aggregator (port_attached is TRUE), the Aggregation Port is operational (Port_Operational is TRUE), the wait-to-restore timer has expired (wtrWhile == 0), and the Aggregation Port is revertive (Wtr_Revertive is TRUE or Wtr_Waiting is FALSE), then a possible sequence of events is for the Mux machine to enter the ATTACHED state and then progress sequentially to the COLLECTING and COLLECTING_DISTRIBUTING states as Partner_Oper_Port_State.Synchronization and then Partner_Oper_Port_State.Collecting become TRUE. If, however, Partner_Oper_Port_State.Synchronization is TRUE when the machine would enter the ATTACHED state, the machine goes directly to the COLLECTING state (if coupled_control is FALSE) or the COLLECTING_DISTRIBUTING state (if coupled_control is TRUE). Similarly, if both Partner_Oper_Port_State.Synchronization and Partner_Oper_Port_State.Collecting are TRUE when the machine would enter the ATTACHED state, the machine goes directly to the COLLECTING_DISTRIBUTING state. The machine returns to the ATTACHED state from the COLLECTING or COLLECTING_DISTRIBUTING states if Partner_Oper_Port_State.Synchronization becomes FALSE. The machine returns to the COLLECTING state from the COLLECTING_DISTRIBUTING state if coupled_control is FALSE and Partner_Oper_Port_State.Collecting becomes FALSE while Partner_Oper_Port_State.Synchronization remains TRUE.

In the ATTACHED state, the value of Actor_Oper_Port_State.Synchronization is set to TRUE. Collection of frames from the Aggregation Port and distribution of frames to the Aggregation Port are disabled, and Actor_Oper_Port_State.Collecting and Actor_Oper_Port_State are set to FALSE.

In the COLLECTING state, collection of frames from the Aggregation Port is enabled, and Actor_Oper_Port_State.Collecting is set to TRUE. Distribution of frames to the Aggregation Port is disabled, and Actor_Oper_Port_State.Distributing is set to FALSE. Actor_Oper_Port_State.Synchronization is always TRUE while in the COLLECTING state.

In the COLLECTING_DISTRIBUTING state, the distribution of frames to the Aggregation Port and collection of frames from the Aggregation Port are both enabled, and Actor_Oper_Port_State.Collecting and Actor_Oper_Port_State.Distributing are set to TRUE. Actor_Oper_Port_State.Synchronization is always TRUE while in the COLLECTING_DISTRIBUTING state.

The sequence of operations and transitions defined for the COLLECTING and COLLECTING_DISTRIBUTING states in this state machine when coupled_control is FALSE ensures that frames are not distributed to an Aggregation Port until the Partner has enabled collection and that distribution is stopped as soon as the Partner's state indicates that collection has been disabled. This sequence minimizes the possibility that frames will be lost or misdelivered during the process of bringing the Aggregation Port into operation or taking the Aggregation Port out of operation. When coupled_control is TRUE, indicating that independent control is not possible, the state machine does not wait for the Partner to signal that collection has started before enabling both collection and distribution.

The NTT variable is set to TRUE in the DETACHED, ATTACHED, COLLECTING, and COLLECTING_DISTRIBUTING states in order to ensure that the Partner is made aware of the changes in the Actor's state variables that are caused by the operations performed in those states.

NOTE—The NTT variable is not set to TRUE in the ATTACHED_WTR state because this state is entered when Port_Operational is FALSE and an LACPDU could not be transmitted. When Port_Operational becomes TRUE, an LACPDU is transmitted by the LACP Transmit machine within a Short_Timeout_Time.

### 6.4.14 LACP Transmit machine

The LACP Transmit machine shall implement the function specified in Figure 6-17 with its associated parameters (6.4.4 through 6.4.10).

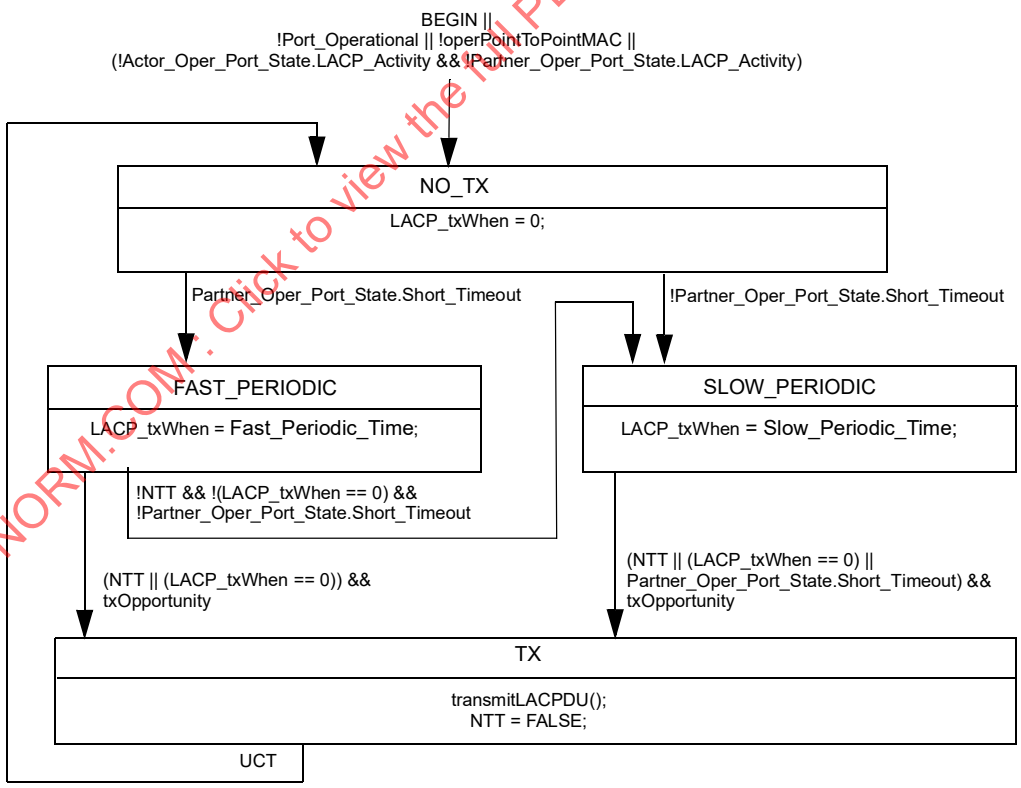



**Figure 6-17—LACP Transmit state diagram**

The LACP Transmit state machine has four states:

a) *NO_TX*. While in this state, LACPDU transmissions are disabled. When Port_Operational is TRUE and either the actor or partner state indicates Active LACP operation, the state machine transitions to either FAST_PERIODIC or SLOW_PERIODIC depending on the receive timeout rate in use by the Partner (indicated by Partner_Oper_Port_State.Short_Timeout).
b) *FAST_PERIODIC*. While in this state, periodic LACPDU transmissions are enabled at a fast transmission rate.
c) *SLOW_PERIODIC*. While in this state, periodic LACPDU transmissions are enabled at a slow transmission rate.
d) *TX*. This is a transitory state entered when NTT has been asserted or the periodic transmission timer (LACP_txWhen) has expired. A single LACPDU is transmitted. Then the state machine transitions through NO_TX to either FAST_PERIODIC or SLOW_PERIODIC.

## 6.5 Marker protocol

### 6.5.1 Introduction

The Marker protocol allows the Frame Distribution function of an Actor's Link Aggregation sublayer to request the transmission of a Marker PDU on a given link. The Marker PDU is received by the Partner's Frame Collection function and a Marker Response PDU is returned on the same link to the initiating Actor's Frame Distribution function. Marker and Marker Response PDUs are not prioritized relative to the frames of the conversations that they mark. Marker/Marker Response PDUs are subject to the operation of flow control, where supported on the link. Hence, if the Frame Distribution function requests transmission of a Marker PDU on a given link and does not transmit any further frames that relate to a given set of conversations until the corresponding Marker Response PDU is received from that link, then it can be certain that there are no frames related to those conversations still to be received by the Partner's Frame Collection function. The use of the Marker protocol can therefore allow the Frame Distribution function a means of determining the point at which a given set of conversations can be reallocated from one link to another without the danger of causing frames in those conversations to be misordered at the Frame Collector.

NOTE—The use of the Marker protocol is further discussed in Annex B. An alternative to the Marker protocol is defined in 6.6.

The operation of the Marker protocol is unaffected by any changes in the Collecting and Distributing states associated with the Aggregation Port. Therefore, Marker and Marker Response PDUs can be sent on an Aggregation Port whose Frame Distribution function is disabled; similarly, such PDUs can be received and passed to the relevant Aggregator's Frame Collection or Distribution function on an Aggregation Port whose Frame Collection function is disabled.

The use of the Marker protocol is optional; however, the ability to respond to Marker PDUs, as defined for the operation of the Marker Responder (see 6.5.4.1 and 6.5.4.2) is mandatory. Some distribution algorithms might not require the use of a marker; other mechanisms (such as timeouts) can be used as an alternative.

The Marker protocol does not provide a guarantee of a response from the Partner; no provision is made for the consequences of frame loss or for the failure of the Partner System to respond correctly. Implementations that make use of this protocol have to therefore make their own provision for handling such cases.

### 6.5.2 Sequence of operations

Figure 6-18 illustrates the sequence of marker operations between an initiating and responding System. Time is assumed to flow from the top of the diagram to the bottom.

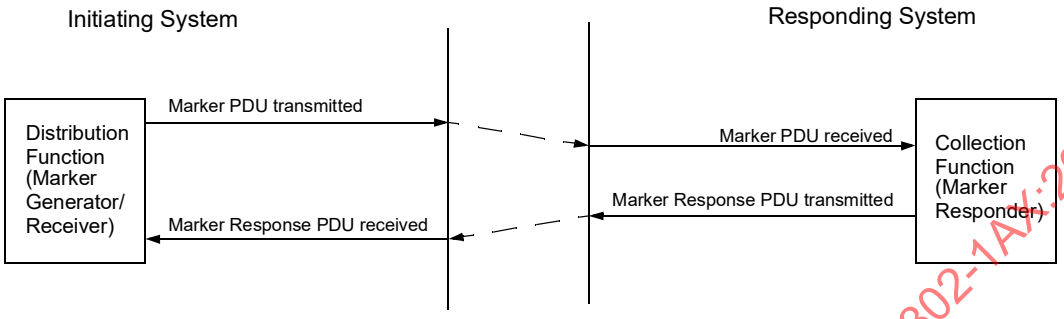



**Figure 6-18—Marker protocol time sequence diagram**

### 6.5.3 Marker and Marker Response PDU structure and encoding

#### 6.5.3.1 Transmission and representation of octets

All Marker and Marker Response PDUs comprise an integral number of octets. The octets in a PDU are numbered starting from 1 and increasing in the order they are put into the PDU. The bits in each octet are numbered from 1 to 8, where bit 1 is the least significant bit.

When consecutive bits within an octet are used to represent a binary number, the highest bit number has the most significant value. When consecutive octets are used to represent a binary number, the most significant octet has the lowest octet number and is transmitted first, followed by successively less significant octets (with successively higher octet numbers).

When the encoding of (an element of) a Marker or Marker Response PDU is depicted in a diagram,

a) Elements of a PDU are transmitted from top to bottom.
b) When an element contains multiple octets, the most significant octet is transmitted first.
c) When multiple octets are depicted side by side, the octets are transmitted from left to right.
d) Within an octet, bits are shown with bit 8 (the most significant bit) to the left and bit 1 (the least significant bit) to the right.

#### 6.5.3.2 Encapsulation of Marker and Marker Response PDU in frames

Marker and Marker Response PDUs are encoded in the mac_service_data_unit parameter of an M_UNITDATA.request or M_UNITDATA.indication. The first octets of the mac_service_data_unit are the Slow Protocols EtherType (Table 6-3), followed by the PDU as specified in 6.5.3.3.

### 6.5.3.3 Marker and Marker Response PDU structure

The Marker PDU and Marker Response PDU structure shall be as shown in Figure 6-19 and further described in the field definitions after the figure.

| Marker PDU | Octets | Marker Response PDU |
|---|---|---|
| Subtype = Marker | 1 | Subtype = Marker |
| Version Number | 1 | Version Number |
| TLV_Type = Marker Information | 1 | TLV_Type = Marker Response Information |
| Marker_Information_Length= 16 | 1 | Marker_Response_Information_Length = 16 |
| Requester_Port | 2 | Requester_Port |
| Requester_System | 6 | Requester_System |
| Requester_Transaction_ID | 4 | Requester_Transaction_ID |
| Pad = 0 | 2 | Pad = 0 |
| TLV_Type = Terminator | 1 | TLV_Type = Terminator |
| Terminator_Length = 0 | 1 | Terminator_Length = 0 |
| Reserved | 90 | Reserved |
| FCS | 4 | FCS |

**Figure 6-19—Marker PDU and Marker Response PDU structure**

a) *Subtype.* The Subtype field identifies the specific Slow Protocol being encapsulated. Both Marker and Marker Response PDUs carry the Marker_subtype value 0x02.

b) *Version number.* This identifies the Marker protocol version; implementations conformant to this version of the standard carry the value 0x01.

c) *TLV_Type = Marker Information/Marker Response Information.* This indicates the nature of the information carried in this TLV-tuple. Marker Information is encoded as the integer value 0x01; Marker Response Information is encoded as the integer value 0x02.

d) *Marker_Information_Length/Marker_Response_Information_Length.* This field indicates the length (in octets) of this TLV-tuple. Both Marker and Marker Response information use a length value of 16 (0x10).

e) *Requester_Port.* The Port Number assigned to the Aggregation Port by the Requester (the System sending the initial Marker PDU).

f) *Requester_System.* The MAC address component of the Requester's System Identifier.

g) *Requester_Transaction_ID*. The transaction ID allocated to this request by the requester, encoded as an integer.

h) *Pad.* This field is used to align TLV-tuples on 16-byte memory boundaries. It is transmitted as zeros in Marker PDUs; in Marker Response PDUs, this field can be transmitted as zeros or with the contents of this field from the Marker PDU triggering the response. The field is ignored on receipt in all cases.

NOTE 1—The difference in handling of the Pad field in Marker Response PDUs allows an implementation to reflect the contents of the received Marker PDU in its response, without enforcing the requirement to transmit the field as zeros.

i) *TLV_Type = Terminator.* This field indicates the nature of the information carried in this TLV-tuple. Terminator (end of message) information carries the integer value 0x00.

j) *Terminator_Length.* This field indicates the length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).

k) *Reserved*. These 90 octets are reserved for use in future extensions to the protocol. It is transmitted as zeros in Marker PDUs; in Marker Response PDUs, this field can be transmitted as zeros or with the contents of this field from the Marker PDU triggering the response. The field is ignored on receipt in all cases.

NOTE 2—These trailing reserved octets are included in all Marker and Marker Response PDUs in order to force a fixed PDU size, regardless of the version of the protocol. Hence, a Version 1 implementation is guaranteed to be able to receive version N PDUs successfully, although version N PDUs can contain additional information that cannot be interpreted (and is ignored) by the Version 1 implementation. A crucial factor in ensuring backwards compatibility is that any future version of the protocol is required not to redefine the structure or semantics of information defined for the previous version; it can add only new information elements to the previous set. Hence, in a version N PDU, a Version 1 implementation can expect to find the Version 1 information in exactly the same places as in a Version 1 PDU and can expect to interpret that information as defined for Version 1.

l) *FCS.* This field is the frame check sequence (FCS), typically generated by the underlying MAC.

### 6.5.4 Protocol definition

#### 6.5.4.1 Operation of the marker protocol

Marker PDUs can be generated by the Frame Distribution function to provide a sequence marker in the stream of frames constituting a conversation or set of conversations. Received Marker PDUs are delivered to the Marker Responder within the Frame Collection function of the Partner System.

On receipt of a valid Marker PDU, the Frame Collection function issues a Marker Response PDU, in the format specified in Figure 6-19, to the same Aggregation Port from which the Marker PDU was received. The Requester_Port, Requester_System, and Requester_Transaction_ID parameter in the Marker Response PDU carry the same values as those received in the corresponding Marker PDU.

Received Marker Response PDUs are passed to the Marker Receiver within the Frame Distribution function. Implementation of the Marker Generator and Receiver is optional.

The Marker Generator, if implemented, shall comply with the frame rate limitation constraint for Slow Protocols, as specified in 57A.3 of IEEE Std 802.3-2018. A Marker Responder can (but is not required to) generate Marker Responses to Marker messages not in compliance with this constraint.

NOTE 1—It is important that Marker Response PDUs not be queued indefinitely, and sent long after the corresponding Marker PDU that triggered the response.

Frames generated by the Marker Responder do not count towards the rate limitation constraint for Slow Protocols, as specified in 57A.3 of IEEE Std 802.3-2018.

NOTE 2—The Marker Responder behaves the same whether Link Aggregation is employed in a single system or as part of a DRNI (Clause 9). In the latter case, frames in transit between one DRNI System and another when a Marker Response is generated can be delivered out of order. See 6.6 for an alternative method of ensuring frame ordering.

### 6.5.4.2 Marker Responder state diagram

The Marker Responder shall implement the function specified in Figure 6-20, with its associated parameters (6.5.4.2.1 and 6.5.4.2.2).

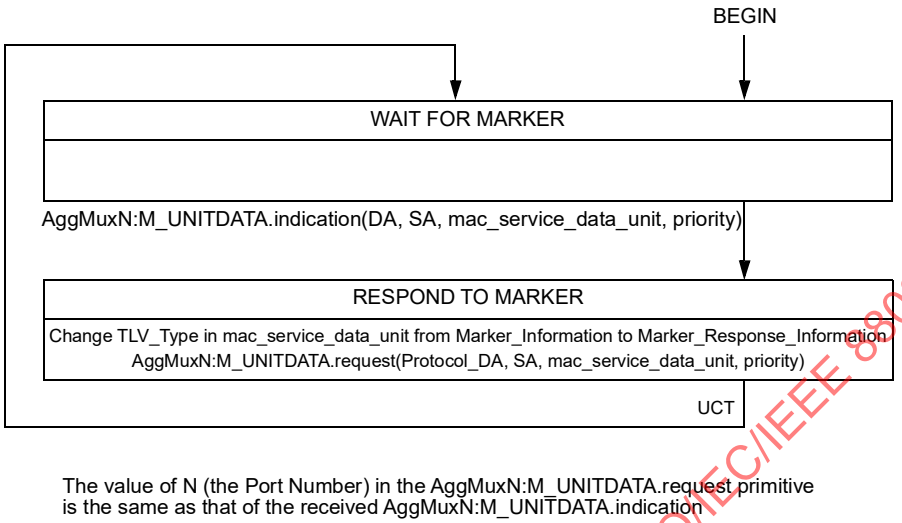



**Figure 6-20—Marker Responder state diagram**

#### 6.5.4.2.1 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.request and M_UNITDATA.indication primitives.

Protocol_DA
: The address determined by the setting of the aAggPortProtocolDA managed object (7.3.2.2.1). Data Type: 6 octets.

#### 6.5.4.2.2 Messages

AggMuxN:M_UNITDATA.request
: The service primitive used to transmit a frame with the specified parameters.

AggMuxN:M_UNITDATA.indication
: The service primitive used to pass a received frame to a client with the specified parameters.

Upon receipt of an AggMuxN:M_UNITDATA.indication primitive, the Marker Responder shall not validate the *Version Number*, *Pad*, or *Reserved* fields in the contained Marker Request PDU. The same actions are taken regardless of the values received in these fields. A Marker Responder may validate the *Marker_Information_Length* field. These behaviors, together with the constraint on future protocol enhancements discussed in the note in item k) of 6.5.3.3, allow Version 1 devices to be compatible with future revisions of the protocol.

## 6.6 Conversation-Sensitive Collection and Distribution

Conversation-Sensitive Collection and Distribution (CSCD) allows administrative control of the Frame Distributor's selection of the Aggregation Link for each frame and allows the Collector to accept frames received only on the expected Aggregation Link.

The distribution of frames can be controlled regardless of whether the Link Aggregation partner supports CSCD. If the partner has a matching CSCD configuration, each conversation supported by the LAG can be distributed to the same Aggregation Link in both directions. This facilitates load balancing, detailed traffic management and bandwidth control, and implementation of monitoring and policing functions on a LAG.

Conversation-Sensitive Collection and Distribution does not require a particular distribution algorithm; rather it creates a structure for identifying and configuring a variety of standardized or vendor-defined distribution algorithms. It uses the following:

a) A Port Algorithm (identified by the Aggregator's Actor_Port_Algorithm variable and, in some cases, supplemented by the Admin_Conv_Service_Map variable) used to associate each frame transmitted or potentially received by an Aggregator with a 12-bit Port Conversation ID (6.6.1).
b) A Collection_Conversation_Mask for each Aggregation Port, with a single bit for each Port Conversation ID that permits or denies reception from the port's Aggregation Link.
c) A Distribution_Conversation_Mask for each Aggregation Port, with a single bit for each Port Conversation ID that permits or denies distribution to the port's Aggregation Link.

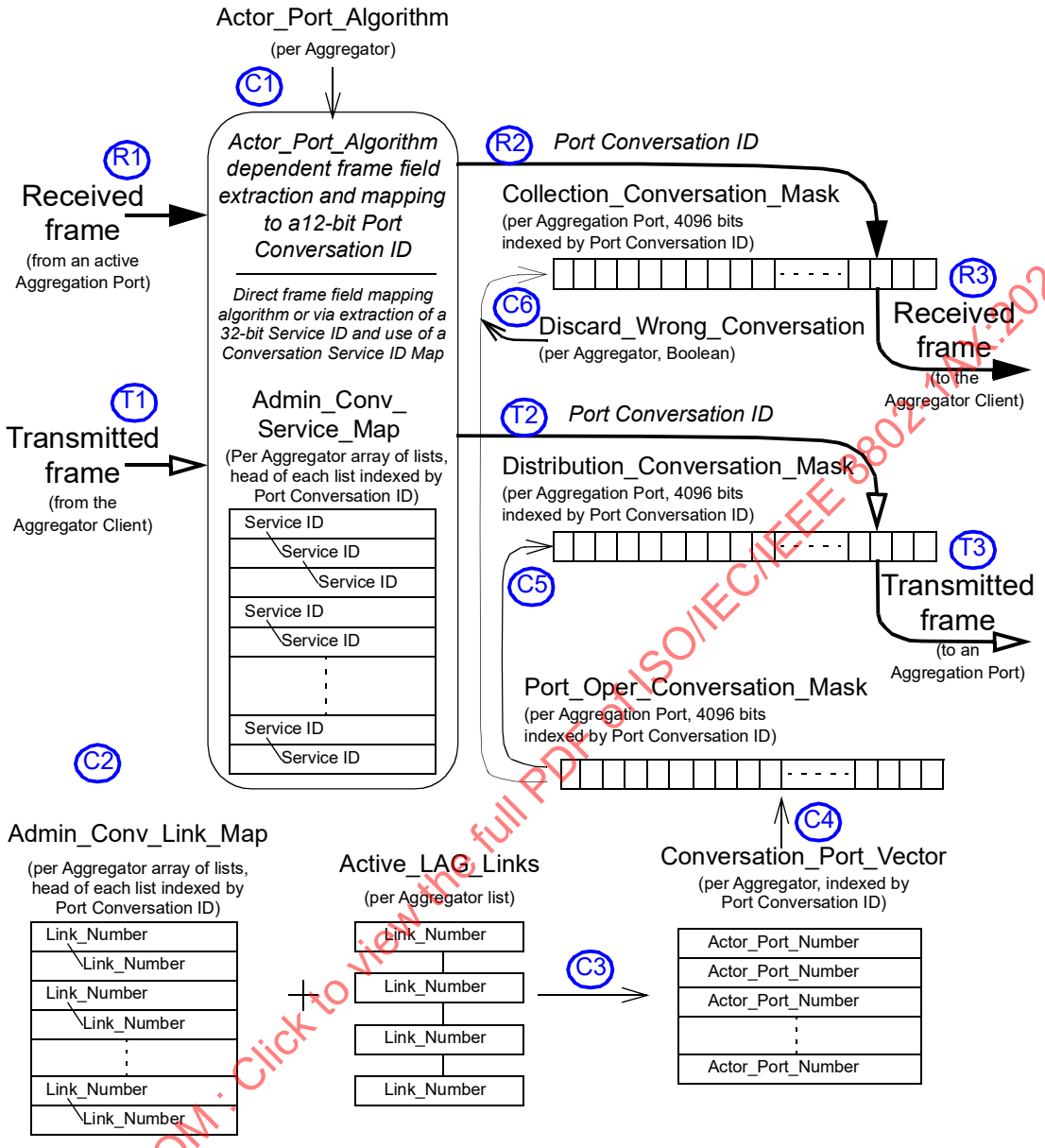
The Collection_Conversation_Mask and Distribution_Conversation_Mask for each Aggregation Port attached to an Aggregator are automatically updated as a consequence of the configuration of, or changes to, the following:

d) The value of Actor_Oper_Port_State.Collecting for any Aggregation Port attached to the Aggregator. The Aggregation Port is considered "active" on the Aggregator when Actor_Oper_Port_State.Collecting is TRUE.
e) The operational Link_Number for each active Aggregation Port attached to the Aggregator.
f) The Admin_Conv_Link_Map for the Aggregator, which determines one (or none) of the active Link Aggregation Ports (identified by their Link_Numbers) to distribute frames for any given Port Conversation ID.
g) The Discard_Wrong_Conversation variable for the Aggregator, which determines whether frames with a given Port Conversation ID are received from any active Aggregation Port or only from the Aggregation Port to which that Port Conversation ID would be distributed.

The Admin_Conv_Link_Map and active Link_Numbers are used to calculate a Conversation_Port_Vector for the Aggregator to map each Port Conversation ID to an Aggregation Port. The Conversation_Port_Vector is then used to generate a Port_Oper_Conversation_Mask for each of the Aggregator's active Aggregation Ports, with a single bit for each Port Conversation ID. The bit is set (TRUE) if frames associated with the Port Conversation ID are to be distributed to that Aggregation Port.

When an Aggregation Port is not active, its Distribution_Conversation_Mask is cleared so that transmission is inhibited; otherwise, it is updated by using its Port_Oper_Conversation_Mask with care not to permit transmission of an conversation through more than one Aggregation Port at the same time.

The operation of Conversation-Sensitive Collection and Distribution is summarized in Figure 6-21. Some details of the operation are necessarily omitted. Figure 6-21 is provided for the convenience of the reader; however, the detailed specification in 6.6.3 takes precedence.




Receive:
**R1:** Frame received from an Aggregation Port that is active in the LAG.
**R2:** The Actor_Port_Algorithm, possibly with the Admin_Conv_Service_Map, determine the frame's Port Conversation ID.
**R3:** The Port Conversation ID is used to look up a Boolean in the Collection_Conversation_Mask. If TRUE (and if the port is collecting) the frame is passed to the Aggregator Client.

Transmit:
**T1:** Frame transmitted by an Aggregation Client.
**T2:** The Actor_Port_Algorithm, possibly with the Admin_Conv_Service_Map, determine the frame's Port Conversation ID.
**T3:** The Port Conversation ID is used to look up a Boolean in the Distribution_Conversation_Mask. The frame is transmitted at an Aggregation Port where the Boolean is TRUE (and the port is distributing).

Control:
**C1**: The Actor_Port_Algorithm (and possibly Conversation_Port_Vector) is configured for each Aggregator.
**C2:** The Admin_Conv_Link_Map is configured for each Aggregator.
**C3:** The Admin_Conv_Link_Map together with the operational list of links active on the Aggregator determine the Aggregation Port to be used for frames associated with each Port Conversation ID.
**C4:** The Conversation_Port_Vector is used to create a Boolean mask for each Aggregation Port in the LAG.
**C5:** The mask is copied to the Distribution_Conversation_Mask to control the distribution by Port Conversation ID.
**C6:** The mask, together with Discard_Wrong_Conversation, control the collection by Port Conversation ID.

**Figure 6-21—Conversation-Sensitive Collection and Distribution overview**

CSCD-capable Systems use version 2 LACP to exchange the Actor's and Partner's Port Algorithm and digests of their Admin_Conv_Service_Map and Admin_Conv_Link_Map and to agree on a common Link_Number for each Aggregation Port; as a result, every Actor can know if its Partner distributes (or is about to distribute) the same conversations to the same links. If so, and if the Aggregator's Admin_Discard_Wrong_Conversation variable value is AUTO, each active Aggregation Port's Port_Oper_Conversation_Mask is used to update its Collection_Conversation_Mask. If the Actor's and Partner's distribution configuration differ, or the Admin_Discard_Wrong_Conversation value is FORCE_FALSE, the Collection_Conversation_Masks are updated to permit reception of any conversation on any of the Aggregation Ports. The Admin_Discard_Wrong_Conversation value can also be FORCE_TRUE, with the Port_Oper_Conversation_Mask used to update the Collection_Conversation_Mask independently of Port Algorithm or Admin_Conv_Link_Map differences. Note that CSCD-capable systems are not required to support the values AUTO and FORCE_TRUE for the Admin_Discard_Wrong_Conversation variable (5.3.2 and 7.3.1.1.35).

### 6.6.1 Port Algorithms and Port Conversation IDs

A Port Algorithm specifies how each frame (i.e., service requests from the Aggregator Port and service indications from the Aggregation Port) is associated with a Port Conversation ID taking a value between 0 and 4095. In particular the algorithm specifies how the contents of one or more fields in the frame are used to determine the Port Conversation ID.

The Port Algorithm is identified by the Actor_Port_Algorithm variable. The three most significant octets of the Actor_Port_Algorithm are an OUI identifying the organization responsible for the specification of the algorithm. The fourth octet differentiates between algorithms specified by the same organization. Subclause 8.2 specifies the algorithms defined by this standard.

Possible algorithms map a frame field (e.g., a VLAN identifier) directly to the Port Conversation ID or use an algorithmic mapping (e.g., a hash algorithm that reduces the contents of specified fields to a 12-bit integer). This standard also supports table-driven mappings (e.g., of an I-SID to a Port Conversation ID) and algorithmic and table-driven mapping combinations, by specifying an optional per Aggregator configurable Admin_Conv_Service_Map that maps 32-bit Service-IDs to Port Conversation IDs. A CSCD-capable Actor and its Partner can exchange an MD5 digest of their Admin_Conv_Service_Map as well as their Actor_Port_Algorithm identifier to confirm that they are using the same frame to Port Conversation ID mapping.

### 6.6.2 Link numbers and link selection

In a CSCD-capable System, each Aggregator's Admin_Conv_Link_Map uses Link_Numbers to identify Aggregation Links, with a local mapping of the Link_Numbers to the Actor_Port_Numbers used to identify Aggregation Ports within the System as a whole. Each Aggregation Port has a locally configured Admin_Link_Number that is unique among the Aggregation Ports with the same Aggregation Key (because they could be attached to the same Aggregator). The Admin_Link_Number is initially used as the operational Link_Number. The operational Link_Number value is transmitted in Version 2 LACPDUs.

If all of the following conditions apply:

a) An Aggregation Port is in current receipt (i.e., Actor_Oper_Port_State.Defaulted and Actor_Oper_Port_State.Expired are both FALSE) of a Version 2 LACPDU;

b) The Aggregation Port is active as described in item d) of 6.6;

c) The concatenation of the Partner's System Identifier (6.3.2) and Port Identifier (6.3.4) is numerically lower than the concatenation of the Actor's System Identifier and Port Identifier;

d) The Actor and its Partner are using the same Port algorithm (with the same Admin_Conv_Service_Map if used by the algorithm) and the same Admin_Conv_Link_Map (i.e., differPortAlgorithms, differConvServiceDigests, and differConvLinkDigests are all FALSE),

then the operational Link_Number used by the Aggregation Port is changed to the value transmitted by the Actor's Partner (Partner_Link_Number); otherwise, the Aggregation Port continues to use the Admin_Link_Number as the operational Link_Number.

The Admin_Conv_Link_Map variable is configured with a link selection preference list, i.e., a sequence of one or more Link_Numbers in preference order (highest to lowest), for each Port Conversation ID. The first Link_Number in the list associated with a currently active Aggregation Port determines the Aggregation Port to which frames associated with that Port Conversation ID are distributed. The current association of Port Conversation IDs to Aggregation Ports is stored in the Aggregator's Conversation_Port_Vector variable, from which each Aggregation Port's Port_Oper_Conversation_Mask is derived.

### 6.6.3 Conversation-sensitive LACP

Conversation-sensitive LACP enables configuration, control, and coordination of the Conversation-Sensitive Collection and Distribution functions of each participating System or DRNI by using static information local to each System or DRNI and dynamic information exchanged using the Version 2 TLVs specified in 6.4.2.4.

For each Aggregator in a System or DRNI, Conversation-sensitive LACP

a) Maintains the configuration information for Conversation-Sensitive Collection and Distribution;
b) Maintains an operational Link_Number list for each Aggregation Port active on the LAG.
c) Uses the configuration information and the list of active links to generate a map of Port Conversation IDs to Aggregation Ports for use by the Distributor.
d) Exchanges configuration information with the Partner System on each Aggregation Port attached to the Aggregator.
e) Uses information from the Partner System's Link Aggregation Control entity to enable or disable per Port Conversation ID frame collection and distribution at the Aggregator Parser/Multiplexer of each Aggregation Port.

The operation of Conversation-sensitive LACP is described in detail in 6.6.3.1 through 6.6.3.5. The Conversation-sensitive LACP is integrated with the Link Aggregation Control in Figure 6-3. Much of the actual operation of Conversation-sensitive LACP is the result of Conversation-sensitive LACP functions (6.6.3.4) invoked by the LACP Receive machine (6.4.11) or Mux machine (6.4.13). The remainder of Conversation-sensitive LACP operation is the result of the Update Mask machine (6.6.3.5).

The parameters that characterize a distribution algorithm come in sets of three, distinguished by the suffixes Port_Algorithm, Conv_Service_Digest, and Conv_Link_Digest. Conversation-sensitive LACP maintains four variations of these parameter sets, distinguished by the prefixes Actor, Actor_Oper, Partner_Oper, and Partner (see Figure 6-22). The use of each of these four sets is best understood by considering the general flow of distribution algorithm information in Conversation-sensitive LACP:

1) An Aggregator's Actor variables are either configured values (Actor_Port_Algorithm) or MD5 digests calculated from configured values (Actor_Conv_Service_Digest and Actor_Conv_Link_Digest calculated from Admin_Conv_Service_Map and Admin_Conv_Link_Map, respectively) that describe the distribution algorithm currently in use by the Aggregator.

2) An Aggregation Port's Actor_Oper variables are copied from the Aggregator's set of Actor variables when the Aggregation Port is attached to the Aggregator. These are conveyed in LACPDUs and accurately reflect the values in use by the actor's Aggregator when Actor_Oper_Port_State.Synchronization is TRUE.
3) An Aggregation Port's Partner_Oper variables are captured from LACPDUs and accurately reflect the values in use by the partner's Aggregator when Partner_Oper_Port_State.Synchronization is TRUE.
4) An Aggregator's Partner_ variables are copied from the Aggregation Port's Partner_Oper variables when both Actor_Oper_Port_State.Synchronization and Partner_Oper_Port_State.Synchronization are TRUE, and then the Mux machine enters the COLLECTING or COLLECTING_DISTRIBUTING state. The Partner variables describe the distribution algorithm believed to be currently in use by the partner's Aggregator.

The Aggregator's Actor variables are compared to the Partner variables, with the result determining the operational values of the Aggregator's Discard_Wrong_Conversation variable and each active Aggregation Port's operational Link_Number variable.

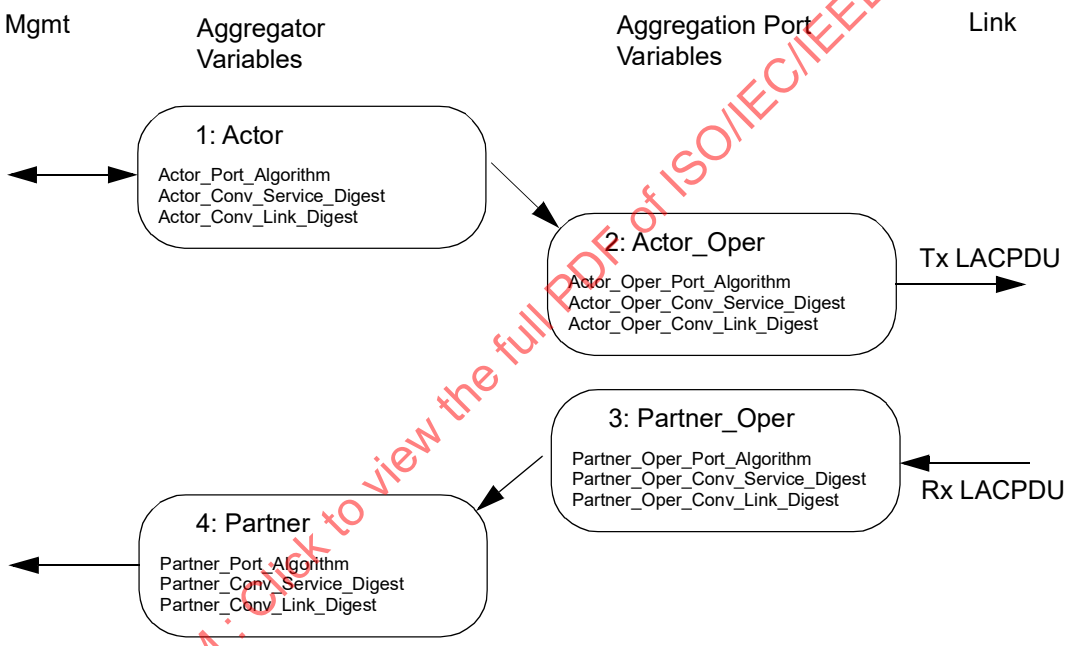



**Figure 6-22—Distribution algorithm information flow**

### 6.6.3.1 Per-Aggregator variables

Actor_Port_Algorithm

This administrative variable indicates the distribution algorithm used by the Actor to assign frames to Port Conversation IDs and is exchanged with the Partner System to determine whether the Partner is using the same distribution algorithm. When modified, the changeActorDistAlg flag is set to TRUE.

Data Type: 4-octet distribution algorithm (Figure 8-1).

Alias of aAggPortAlgorithm (7.3.1.1.33).

Admin_Conv_Service_Map

An array, indexed by Port Conversation ID, of entries, where each entry is a set of Service IDs (8.2) that map to that Conversation ID. Any given Service ID value can appear in at most one entry in the array. A Service ID appearing in more than one entry in the array can, when

Discard_Wrong_Conversation is TRUE, cause significant frame loss. A Service ID value not contained in any entry does not map to any Conversation ID, and frames associated with that Service ID value are discarded. When modified, the Actor_Conv_Service_Digest is recalculated, and, if the most significant bit of the fourth octet of Actor_Port_Algorithm is 1 indicating distribution algorithm uses the Admin_Conv_Service_Map, the changeActorDistAlg flag is set to TRUE.
Data Type: Array, indexed by Conversation ID, of lists of Service IDs (8.2).
Alias of aAggAdminConvServiceMap (7.3.1.1.36).

Actor_Conv_Service_Digest
A digest of Admin_Conv_Service_Map for exchange with the Partner System to determine whether the Partner is using the same map. The digest is a 16-octet MD5 fingerprint (IETF RFC 1321) and is calculated whenever there is an administrative change to the Admin_Conv_Service_Map. To calculate the digest, Admin_Conv_Service_Map is considered to contain 4096 consecutive elements, where each element contains a list of zero or more Service IDs (8.2), encoded as binary numbers in increasing order, followed by the Port Conversation ID to which they are mapped. The first element contains the Service ID(s) assigned to Port Conversation ID 0, the second element contains the Service ID(s) assigned to Port Conversation ID 1, the third element contains the Service ID(s) assigned to Port Conversation ID 2, and so on, with the last element containing the Service ID(s) assigned to Port Conversation ID 4095.
Data Type: MD5 digest.
Alias of aAggConvServiceDigest (7.3.1.1.39).

Admin_Conv_Link_Map
An array, indexed by Port Conversation ID, of link selection preference lists. A link selection preference list is a sequence of one or more Link_Numbers in the order of preference, highest to lowest, for the corresponding link to carry that Port Conversation ID. The first value in the sequence that is the same as a Link_Number in the Active_LAG_Links list means frames with that Port Conversation ID will be distributed to the Aggregation Port with that operational Link_Number. A Link_Number value of zero is used to indicate that no link is assigned to carry the associated Port Conversation ID. When modified, the Actor_Conv_Link_Digest is recalculated, and the changeActorDistAlg and changeConvLinkMap flags are set to TRUE.
Data Type: Array, indexed by Conversation ID, of lists of Link_Numbers.
Alias of aAggAdminConvLinkMap (7.3.1.1.34).

Actor_Conv_Link_Digest
A digest of Admin_Conv_Link_Map for exchange with the Partner System to determine whether the Partner is using the same map. The digest is a 16-octet MD5 fingerprint (see IETF RFC 1321) and is calculated whenever there is an administrative change to the Admin_Conv_Link_Map. To calculate the digest, Admin_Conv_Link_Map is considered to contain 4096 consecutive elements, where each element contains a list of Link_Numbers, encoded as binary numbers in the order of preference, highest to lowest, followed by the Port Conversation ID. The first element contains the prioritized list of Link_Numbers assigned to Port Conversation ID 0, the second element contains the prioritized list of Link_Numbers assigned to Port Conversation ID 1, the third element contains the prioritized list of Link_Numbers assigned to Port Conversation ID 2, and so on, with the last element containing the prioritized list of Link_Numbers assigned to Port Conversation ID 4095.
Data Type: MD5 digest.
Alias of aAggConvLinkDigest (7.3.1.1.38).

Partner_Port_Algorithm
The most recent value of Partner_Oper_Port_Algorithm for any Aggregation Port active on this Aggregator. Set by the compareDistributionAlgorithms function and used by the compareDistributionAlgorithms function of the Update Mask machine.
Data Type: 4-octet distribution algorithm (Figure 8-1).
Alias of aAggPartnerPortAlgorithm (7.3.1.1.40).

Partner_Conv_Service_Digest

The most recent value of Partner_Oper_Conv_Service_Digest for any Aggregation Port active on this Aggregator. Set by the compareDistributionAlgorithms function and used by the compareDistributionAlgorithms function of the Update Mask machine.

Data Type: MD5 digest.

Alias of aAggPartnerConvServiceDigest (7.3.1.1.42).

Partner_Conv_Link_Digest

The most recent value of Partner_Oper_Conv_Link_Digest for any Aggregation Port active on this Aggregator. Set by the compareDistributionAlgorithms function and used by the compareDistributionAlgorithms function of the Update Mask machine.

Data Type: MD5 digest.

Alias of aAggPartnerConvLinkDigest (7.3.1.1.41).

Active_LAG_Links

A list, possibly empty, of the operational Link_Number associated with each Aggregation Port attached to this Aggregator that is active on the LAG (i.e., Actor_Oper_Port_State.Collecting is TRUE). The Update Mask machine creates the list in the updateActiveLinks function and uses the list in the updateConversationPortVector function.

Data Type: A list of Link_Numbers.

Alias of aAggActiveLinks (7.3.1.1.43).

Admin_Discard_Wrong_Conversation

Provides administrative control over the Discard_Wrong_Conversation parameter that determines whether the Frame Collector discards any frame received from an Aggregation Port with a Port Conversation ID not set in Port_Oper_Conversation_Mask. The variable is set equal to aAggAdminDiscardWrongConversation (7.3.1.1.35), and the default value is AUTO. The changeDistAlg flag is set to TRUE if this variable is changed while there are any active Aggregation Ports attached to the Aggregator.

Values:

FORCE_TRUE: The value of Discard_Wrong_Conversation is always TRUE.

FORCE_FALSE: The value of Discard_Wrong_Conversation is always FALSE.

AUTO: The value of Discard_Wrong_Conversation is TRUE if all of differPortAlgorithms, differConvLinkDigests, and differConvServiceDigests are FALSE, and the value is FALSE otherwise.

Alias of aAggAdminDiscardWrongConversation (7.3.1.1.35).

Discard_Wrong_Conversation

When TRUE, the Collection_Conversation_Mask for each Aggregation Port active on the Aggregator is generated from the Port_Oper_Conversation_Mask so that M_UNITDATA.indication primitives received at the Aggregation Ports are discarded or passed to the Frame Collector by the Aggregator Parsers according to the Port_Conversation_ID. When FALSE, the Collection_Conversation_Masks are set to TRUE for all Port_Conversation_IDs so the Aggregator Parsers do not filter M_UNITDATA.indication primitives received at the Aggregation Ports. The value is determined by the compareDistributionAlgorithms function of the Update Mask machine.

Data Type: Boolean.

Alias of aAggOperDiscardWrongConversation (7.3.1.1.37).

Conversation_Port_Vector

An array, indexed by Conversation ID, that contains the Actor_Port_Number of the Aggregation Port attached to this Aggregator that carries frames associated with a given Conversation ID. The Update Mask machine creates the vector in the updateConversationPortVector function and uses the vector in the updateConversationMasks function.

Data Type: sequence of Actor_Port_Numbers.

#### 6.6.3.2 Variables associated with each Aggregation Port

Admin_Link_Number
: The Link_Number value for the Aggregation Port, configured by the System's administrator, that is unique among all Aggregation Ports that have the same Actor_Admin_Port_System_Priority, Actor_Admin_Port_System, and Actor_Admin_Port_Key values and selected from the set of Link_Numbers in the Admin_Conv_Link_Map of any Aggregator with matching Actor_Admin_System_Priority, Actor_Admin_System, and Actor_Admin_Aggregator_Key values. More than one Aggregation Port in a LAG having the same Admin_Link_Number can, if Discard_Wrong_Conversation is TRUE, result in significant frame loss. If the Admin_Link_Number is not in the set of Link_Numbers in the Admin_Conv_Link_Map, then no frames will be distributed to this Aggregation Port. If this value is changed when Actor_Oper_Port_State.Collecting is TRUE, then the selected Aggregator's changeLinkState flag is set to TRUE.
: Data Type: Integer.
: Value: 0 to 65535.
: Alias of aAggPortAdminLinkNumber (7.3.2.1.27).

NOTE—While setting the Admin_Link_Number to 0 is allowed, the consequence is that when this value is used as the operational Link_Number, no frames are distributed to this Aggregation Port.

Link_Number
: This System's operational Link_Number for this Aggregation Port, used to identify the Aggregation Link within the LAG. When the Aggregation Port is collecting, and the Actor and Partner Aggregators have matching distribution algorithm parameters, the value of the Link_Number will be common to both the Actor and Partner; otherwise, the value is local to the Actor and is copied from the Admin_Link_Number.
: Data Type: Integer.
: Value: 0 to 65535.
: Alias of aAggPortOperLinkNumber (7.3.2.1.28).

Partner_Link_Number
: The last-received value of the Partner_Link_Number, or zero if the Aggregator Port is using default values for the Partner or the Partner_LACP_Version is 1.
: Data Type: Integer.
: Value: 0 to 65535.
: Alias of aAggPortPartnerLinkNumber (7.3.2.1.29).

Port_Oper_Conversation_Mask
: The operational Boolean vector, indexed by Port Conversation ID, that indicates whether the indexed Port Conversation ID is distributed through this Aggregation Port (TRUE = passes). This variable is cleared (value set to FALSE for all Conversation IDs) when Actor_Oper_Port_State.Collecting is set to FALSE and maintained by the updateConversationMasks function when Actor_Oper_Port_State.Collecting is TRUE.
: Data Type: sequence of Boolean values, indexed by Port Conversation ID.
: Alias of aAggPortOperConversationPasses (7.3.2.1.25).

Actor_Oper_Conv_Link_Digest
: The prioritized Port Conversation ID-to-Aggregation Port assignments of the Aggregator to which this Aggregation Port was most recently attached. The default value for an Aggregation Port that has not yet been attached to an Aggregator is NULL. This variable is conveyed in Version 2 LACPDUs by the Port Conversation Link Digest TLV (6.4.2.4.2).
: Data Type: MD5 digest.

Actor_Oper_Conv_Service_Digest
: The Port Conversation ID-to-Service ID assignments of the Aggregator to which this Aggregation Port was most recently attached. The default value for an Aggregation Port that

has not yet been attached to an Aggregator is NULL. This variable is conveyed in Version 2 LACPDUs by the optional Port Conversation Service Digest TLV (6.4.2.4.3).
Data Type: MD5 digest.

Actor_Oper_Port_Algorithm
The algorithm to assign frames to Port Conversation IDs used by the Aggregator to which this Aggregation Port was most recently attached. The default value for an Aggregation Port that has not yet been attached to an Aggregator is "Unspecified" (Table 8-1). This variable is conveyed in Version 2 LACPDUs by the Port Algorithm TLV (6.4.2.4.1).
Data Type: 4-octet distribution algorithm (Figure 8-1).

Partner_Oper_Conv_Link_Digest
The most recently received digest of the Partner System's prioritized Port Conversation ID-to-Aggregation Port assignments.
Data Type: MD5 digest.

Partner_Oper_Conv_Service_Digest
The most recently received digest of the Partner System's Port Conversation ID-to-Service ID assignments.
Data Type: MD5 digest.

Partner_Oper_Port_Algorithm
The most recently received value of the algorithm used by the Partner System to assign frames to Port Conversation IDs.
Data Type: 4-octet distribution algorithm (Figure 8-1).

#### 6.6.3.3 Variables used for managing the operation of the state diagrams

BEGIN
This variable indicates the initialization (or reinitialization) of the Conversation-sensitive LACP protocol entity. It is set to TRUE when the System is initialized or reinitialized and is set to FALSE when (re)initialization has completed.
Data Type: Boolean.

The following variables are controlled by the execution of the state machines.

changeLinkState
A signal to an Aggregator's Update Mask machine indicating that there has been a change to the state of an Aggregation Link that could require an update of the Aggregator's Active_LAG_Links variable. It is set to TRUE when there is a change to Actor_Oper_Port_State.Collecting for any Aggregation Port attached to this Aggregator or a change to the Link_Number associated with an Aggregation Port attached to this Aggregator that has Actor_Oper_Port_State.Collecting TRUE. It is also set to TRUE when there is a change to the Admin_Conv_Link_Map or the Admin_Link_Number for this Aggregator when Actor_Oper_Port_State.Collecting is TRUE. It is set to FALSE by the Update Mask state machine.
Data Type: Boolean.

changeAggregationLinks
A signal internal to an Aggregator's Update Mask machine indicating that there has been a change to the list of Link_Numbers associated with Aggregation Ports that are active on the Aggregator or that there has been an administrative change to the Admin_Conv_Link_Map when one or more Aggregation Ports are active on the Aggregator.
Data Type: Boolean.

changeActorDistAlg
A signal that is set to TRUE in response to an administrative change to the Actor_Port_Algorithm, Admin_Conv_Service_Map, or Admin_Conv_Link_Map variable for the Aggregator. It is set to FALSE by the Update Mask state machine.
Data Type: Boolean.

changeConvLinkMap
: A signal to an Aggregator's Update Mask machine indicating that there has been an administrative change to the Admin_Conv_Link_Map.
Data Type: Boolean.

changeDistAlg
: A signal indicating there has been a change to the distribution algorithm parameters while there are one or more active Aggregation Links in the LAG. It is set to TRUE in response to an administrative change to the Actor_Port_Algorithm, Admin_Conv_Service_Map, or Admin_Conv_Link_Map variable for the Aggregator while an Aggregation Port is active or a change to the Partner_Oper_Port_Algorithm, Partner_Oper_Conv_Service_Digest, or Partner_Oper_Conv_Link_Digest variable on an active Aggregation Port. This flag is also set to TRUE when there is a change to the Admin_Discard_Wrong_Conversation variable while there are any active Aggregation Ports attached to the Aggregator. It is set to FALSE by the Update Mask state machine.
Data Type: Boolean.

changeCSCD
: A signal internal to an Aggregator's Update Mask machine indicating that the Aggregator's Conversation_Port_Vector has changed or the current operational value of Discard_Wrong_Conversation has changed on an Aggregator with active Aggregation Links.
Data Type: Boolean.

differConvServiceDigests
: A Boolean indicating that the most significant bit of the fourth octet of both the Actor_Port_Algorithm and Partner_Port_Algorithm is 1 and the Actor_Conv_Service_Digest and the Partner_Conv_Service_Digest are different. Calculated and used in the compareDistributionAlgorithms function of the Update Mask machine.
Data Type: Boolean.

differPortAlgorithms
: A Boolean that, when TRUE, indicates the Actor_Port_Algorithm and the Partner_Port_Algorithm are different. Calculated and used in the compareDistributionAlgorithms function of the Update Mask machine.
Data Type: Boolean.

differConvLinkDigests
: A Boolean indicating that the Actor_Conv_Link_Digest and the Partner_Conv_Link_Digest are different or that the Partner_Link_Number is zero indicating that the most recently received LACPDU did not contain a Port Conversation Link Digest TLV. Calculated and used in the compareDistributionAlgorithms function of the Update Mask machine.
Data Type: Boolean.

#### 6.6.3.4 Functions

recordDefaultPortCSCD
: This function sets the Partner_Link_Number to zero, sets the value of Partner_Oper_Port_Algorithm to "Unspecified" (Table 8-1), and sets Partner_Oper_Conv_Service_Digest and Partner_Oper_Conv_Link_Digest to all zeros.

recordPortCSCD
: If a Port Algorithm TLV (6.4.2.4.1) is included in the received LACPDU, then the value of the *Actor_Port_Algorithm* field in the TLV is stored as the current operational parameter value for Partner_Oper_Port_Algorithm; otherwise, Partner_Oper_Port_Algorithm is set to "Unspecified" (Table 8-1).

  If the most significant bit of the fourth octet of the Partner_Oper_Port_Algorithm is 1 and a Port Conversation Service Digest TLV (6.4.2.4.3) is included in the received LACPDU, then the value for the *Actor_Conv_Service_Digest* field in the TLV is recorded as the current

operational parameter value for Partner_Oper_Conv_Service_Digest. Otherwise, the Partner_Oper_Conv_Service_Digest is set to all zeros, and, if the most significant bit of the fourth octet of the Actor_Oper_Port_Algorithm is 1, then the most significant bit of the fourth octet of the Partner_Oper_Port_Algorithm is set to 0.

If a Port Conversation Link Digest TLV (6.4.2.4.2) is included in the received LACPDU, then the values of the *Link_Number* and *Actor_Conv_Link_Digest* fields in the TLV are stored as the current operational parameter values for the Partner_Link_Number and Partner_Oper_Conv_Link_Digest, respectively. Otherwise, the Partner_Link_Number is set to zero, and the Partner_Oper_Conv_Link_Digest is set to all zeros.

If this results in a change to the Partner_Link_Number value when Actor_Oper_Port_State.Collecting is TRUE, the changeLinkState flag for the selected Aggregator is set to TRUE. If this results in a change to the value of Partner_Oper_Port_Algorithm, Partner_Oper_Conv_Service_Digest, or Partner_Oper_Conv_Link_Digest when Actor_Oper_Port_State.Collecting is TRUE, then the newly recorded values of Partner_Oper_Port_Algorithm, Partner_Oper_Conv_Service_Digest, and Partner_Oper_Conv_Link_Digest are copied to the Aggregator's Partner_Port_Algorithm, Partner_Conv_Service_Digest, and Partner_Conv_Link_Digest variables, respectively; and the changeDistAlg flag is set to TRUE.

resetAggregatorCSCD

This function initializes the Aggregator's Conversation-Sensitive Collection and Distribution variables as follows:

The Partner_Port_Algorithm variable is set to "Unspecified" (Table 8-1), and the Partner_Conv_Link_Digest and Partner_Conv_Service_Digest variables are set to all zeros.

The values of differPortAlgorithms, differConvLinkDigests, and differConvServiceDigests are set to FALSE.

If the value of Admin_Discard_Wrong_Conversation is FORCE_TRUE, then Discard_Wrong_Conversation is set to TRUE; otherwise, Discard_Wrong_Conversation is set to FALSE.

The Active_LAG_Links variable is set to an empty list.

The value of the Conversation_Port_Vector variable is set to zero for all Conversation IDs.

updateActiveLinks

This function builds a new Active_LAG_Links list based on the current state of the Aggregation Ports attached to this Aggregator. The Active_LAG_Links is first emptied. Then, for each Aggregation Port in this Aggregator's LAG_Ports list with Actor_Oper_Port_State.Collecting equal to TRUE, Partner_Oper_Port_State.Synchronization equal to TRUE, and Selected equal to SELECTED,

If the logical 'OR' of the Aggregator's differPortAlgorithms, differConvServiceDigests, and differConvLinkDigests flags is FALSE and if the concatenation of Partner_Oper_System_Priority, Partner_Oper_System, Partner_Oper_Port_Priority, and Partner_Oper_Port_Number is numerically less than the concatenation of Actor_Oper_Port_System_Priority, Actor_Oper_Port_System, Actor_Port_Priority, and Actor_Port_Number, then the value of Partner_Link_Number is copied to Link_Number. Otherwise, the value of Admin_Link_Number is copied to Link_Number.

The resulting Link_Number is placed in the new Active_LAG_Links list.

If the new Active_LAG_Links list is different from the old, the changeAggregationLinks flag is set to TRUE and, if the Aggregator supports a DRNI Gateway, the newHomeInfo value for the DRNI Gateway is set to TRUE.

updateActorDistributionAlgorithm

For each Aggregation Port in the LAG_Ports list, the values of the Aggregator's Actor_Port_Algorithm, Actor_Conv_Service_Digest, and Actor_Conv_Link_Digest variables are copied to the Aggregation Port's Actor_Oper_Port_Algorithm, Actor_Oper_Conv_Service_Digest, and Actor_Oper_Conv_Link_Digest variables, respectively. If any of these Aggregation Ports are active (Actor_Oper_Port_State.Collecting is TRUE), then

The changeDistAlg flag is set to TRUE; and

If the changeConvLinkMap flag is TRUE, the changeAggregationLinks flag is also set to TRUE; and

If the Aggregator supports a DRNI Gateway, the newHomeInfo value for the DRNI Gateway is set to TRUE.

If Actor_Oper_Port_State.Synchronization is TRUE for any port in the LAG_Ports list, then NTT is set to TRUE for that port. Finally the changeConvLinkMap flag is set to FALSE.

updateConversationMasks

For each active Aggregation Port (i.e., Actor_Oper_Port_State.Collecting is TRUE) attached to this Aggregator,

This function computes a new value for the Port_Oper_Conversation_Mask from this Aggregator's Conversation_Port_Vector and the Aggregation Port's Actor_Port_Number. For each Port Conversation ID, the value of the Port_Oper_Conversation_Mask is TRUE if the value in the Conversation_Port_Vector matches the Actor_Port_Number and is FALSE otherwise.

This function then pushes the new value of the Port_Oper_Conversation_Mask to the Distribution_Conversation_Mask of the Aggregator Multiplexer for that Aggregation Port. This operation is performed in a way that assures that the value of the Distribution_Conversation_Mask for a given Port Conversation ID will not be TRUE for more than one Aggregation Port active on this Aggregator at any given time. This can be accomplished by first replacing the value of the Distribution_Conversation_Mask with the logical "AND" of the old Distribution_Conversation_Mask and the new Port_Oper_Conversation_Mask at each Aggregation Port. When all Aggregation Ports active on the Aggregator have been updated in this way, then the new Port_Oper_Conversation_Mask is copied to the Distribution_Conversation_Mask at each Aggregation Port.

When Discard_Wrong_Conversation is TRUE, the new value of the Port_Oper_Conversation_Mask is also pushed to the Collection_Conversation_Mask of the Aggregator Parser for each active Aggregation Port using the process described above. When Discard_Wrong_Conversation is FALSE, the Collection_Conversation_Mask for all Aggregation Ports active on the Aggregator is set to TRUE for all Port Conversation IDs.

updateConversationPortVector

This function updates the mapping of Port Conversation IDs to Actor_Port_Number contained in the Conversation_Port_Vector variable based on the current values of the Admin_Conv_Link_Map and Active_LAG_Links variables. For each Port Conversation ID, the function evaluates the list of Link_Numbers contained in the Admin_Conv_Link_Map and selects the first value that matches a value in the list of Active_LAG_Links. The Actor_Port_Number of the Aggregation Port associated with this Link_Number becomes the entry in the Conversation_Port_Vector for that Port Conversation ID. If no matching value is found, or if the function encounters the value zero in the list from the Admin_Conv_Link_Map

before a matching value is found, then the entry in the Conversation_Port_Vector for that Port Conversation ID is zero.

If this results in a change to the value of the Conversation_Port_Vector variable, the changeCSCD flag is set to TRUE.

compareDistributionAlgorithms

This function compares the values of Partner_Oper_Port_Algorithm, Partner_Oper_Conv_Service_Digest, and Partner_Oper_Conv_Link_Digest for all Aggregation Ports in the Aggregator's LAG_Ports list that have Actor_Oper_Port_State.Collecting TRUE to the Aggregator's Actor_Port_Algorithm, Actor_Conv_Service_Digest, and Actor_Conv_Link_Digest values, and updates the differPortAlgorithms, differConvServiceDigests, differConvLinkDigests, and Discard_Wrong_Conversation values accordingly.

The value of each port's Partner_Oper_Port_Algorithm is compared to the Aggregator's Actor_Port_Algorithm, and if any values are different or any of them has the value "Unspecified" (Table 8-1), then

The Aggregator's Partner_Port_Algorithm is set to "Unspecified," and differPortAlgorithms is set to TRUE.

Otherwise:

The Aggregator's Partner_Port_Algorithm is set to Actor_Port_Algorithm, and differPortAlgorithms is set to FALSE.

The value of each port's Partner_Oper_Conv_Service_Digest is compared to the Aggregator's Actor_Conv_Service_Digest, and if any values are different and the most significant bit of the fourth octet of the Actor_Port_Algorithm is 1, then

The Aggregator's Partner_Conv_Service_Digest is set to any value not matching the Actor_Conv_Service_Digest, and differConvServiceDigests is set to TRUE.

Otherwise:

The Aggregator's Partner_Conv_Service_Digest is set to Actor_Conv_Service_Digest, and differConvServiceDigests is set to FALSE.

The value of each port's Partner_Oper_Conv_Link_Digest is compared to the Aggregator's Actor_Conv_Link_Digest, and if any values are different or if any Partner_Link_Number is zero, then

The Aggregator's Partner_Conv_Link_Digest is set to any value not matching the Actor_Conv_Link_Digest, and differConvServiceDigests is set to TRUE.

Otherwise:

The Aggregator's Partner_Conv_Link_Digest is set to Actor_Conv_Link_Digest, and differConvServiceDigests is set to FALSE.

If any of these actions results in a change to any of the differPortAlgorithms, differConvServiceDigests, or differConvLinkDigests flags, and the Aggregator supports a DRNI Gateway, the newHomeInfo value for the DRNI Gateway is set to TRUE. If this results in a change to the logical 'OR' of the differPortAlgorithms, differConvServiceDigests, and differConvLinkDigests flags and if the concatenation of Partner_Oper_System_Priority, Partner_Oper_System, Partner_Oper_Port_Priority, and Partner_Oper_Port_Number is numerically less than the concatenation of Actor_Oper_Port_System_Priority, Actor_Oper_Port_System, Actor_Port_Priority, and Actor_Port_Number, then the changeLinkState flag is set to TRUE.

If all of differPortAlgorithms, differConvLinkDigests, and differConvServiceDigests are FALSE, or if Admin_Discard_Wrong_Conversation is FORCE_TRUE, then

The variable Discard_Wrong_Conversation is set to TRUE.

Otherwise:

The variable Discard_Wrong_Conversation is set to FALSE.

If this action results in a change to the Discard_Wrong_Conversation value when the Active_LAG_Links list is not empty, then the changeCSCD flag is set to TRUE, and, if the Aggregator supports a DRNI Gateway, the newHomeInfo value for the DRNI Gateway is set to TRUE.

#### 6.6.3.5 Update Mask machine

A System supporting Conversation-Sensitive Collection and Distribution shall implement the Update Mask state machine for each Aggregator as specified in Figure 6-23 with its associated parameters (6.6.3.1 through 6.6.3.4).

The Update Mask machine is responsible for maintaining the Aggregator's Conversation_Port_Vector (that tells the Distributor which Port Conversation IDs are mapped to each Aggregation Port) and each Aggregation Port's Distribution_Conversation_Mask and Collection_Conversation_Mask (that enforce the Conversation-Sensitive Collection and Distribution at the Aggregator Parser/Multiplexer). The flags that initiate action in the Update Mask machine are set by two general classes of events:

a) Enabling or disabling collecting at an Aggregation Port attached to the Aggregator.
b) A management action in the Actor or Partner System that changes a Conversation-Sensitive Collection and Distribution administrative variable when one or more Aggregation Ports attached to the Aggregator are Collecting.

The structure of the Update Mask machine imposes a processing sequence in response to such events.

In response to BEGIN, the Update Mask machine initializes all of the Aggregator's operational variables for Conversation-Sensitive Collection and Distribution and then enters the IDLE state.

UPDATE_ACTOR_ALGORITHMS is entered in response to any management action that results in a change to the Aggregator's Actor_Port_Algorithm, Actor_Conv_Service_Digest, or Actor_Conv_Link_Digest variable. It copies these values to the corresponding operational variables of each Aggregation Port that has selected this Aggregator. If any of these ports have Actor_Oper_Port_State.Collecting TRUE, then the changeDistAlg flag is set to TRUE.

COMPARE_ALGORITHMS is entered in response to any management or protocol action that results in a change to the Actor or Partner distribution algorithm variables, or the Admin_Discard_Wrong_Conversation variable, on an Aggregator that has one or more active Aggregation Ports (any Aggregation Port that has selected this Aggregator has Actor_Oper_Port_State.Collecting TRUE). The Actor and Partner distribution algorithm variables are compared, and the differPortAlgorithms, differConvServiceDigests, and differConvLinkDigests flags are updated. If any of these flags change value when the Partner System ID is numerically lower than the Actor System ID, then changeLinkState is set to TRUE (because the decision of whether to use the Admin_Link_Numbers or Partner_Link_Numbers changes). The Discard_Wrong_Conversation variable is updated, and if it changes, the changeCSCD flag is set.

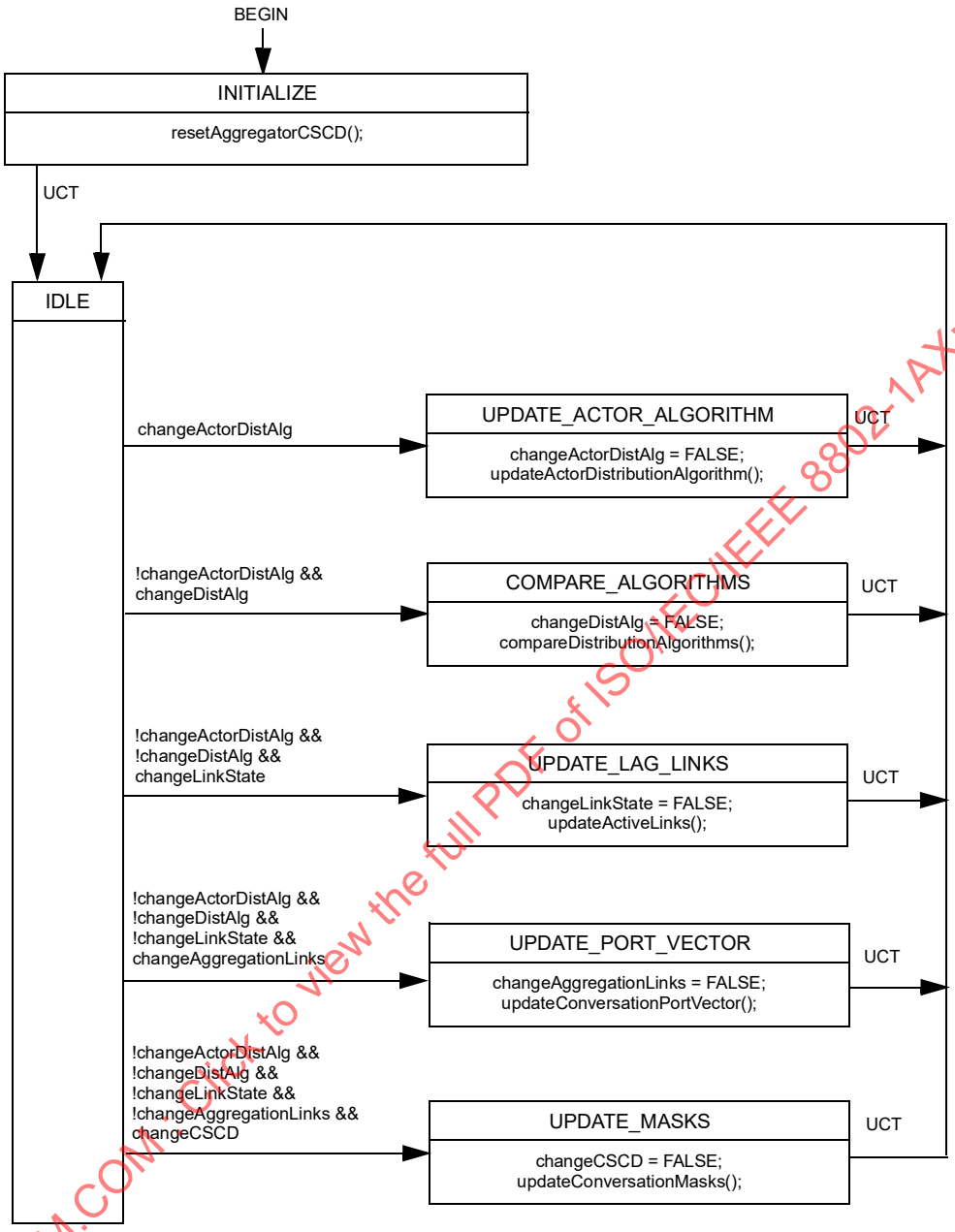



**Figure 6-23—Update Mask state diagram**



UPDATE_LAG_LINKS is entered whenever the value of Actor_Oper_Port_State.Collecting changes on an Aggregation Port that has selected this Aggregator or in response to any management or protocol action that could change the operational Link_Number variable on an Aggregation Port when Actor_Oper_Port_State.Collecting is TRUE. The operational Link_Number is updated for all active Aggregation Ports, and a new list of the Link_Numbers of all active Aggregation Ports (Active_LAG_Links) is generated for the Aggregator. If the new list differs from the old, the changeAggregationLinks flag is set to TRUE.

UPDATE_PORT_VECTOR is entered if the updateActiveLinks() function changes the Active_LAG_Links variable or if there has been an administrative change to the Aggregator's Admin_Conv_Link_Map variable

when any Aggregation Ports are active. A new value for the Conversation_Port_Vector variable (that maps Port Conversation IDs to Actor_Port_Numbers) is generated using the Admin_Conv_Link_Map and Active_LAG_Links list. If the new value of the Conversation_Port_Vector differs from the old, the changeCSCD flag is set to TRUE.

UPDATE_MASKS is entered if there has been a change to the Conversation_Port_Vector or Discard_Wrong_Conversation variable. For each Aggregation Port that is active on the Aggregator, a new Port_Oper_Conversation_Mask is generated from the Conversation_Port_Vector. This is then used to update the Distribution_Conversation_Mask and Collection_Conversation_Mask of the Aggregator Parser/Multiplexer for each Aggregation Port.

NOTE 1—The process used by the updateConversationMasks() function to copy the Port_Oper_Conversation_Mask to the Collection_Conversation_Mask and/or Distribution_Conversation_Mask ensures that at no point in time will the value for any Port Conversation ID be TRUE at more than one Aggregation Port attached to the same Aggregator. This minimizes the risk of any frame duplication or misordering during the update.

NOTE 2—The Update Mask state diagram is detailed so that each function is invoked only when its input parameters have changed. No harm is done by invoking a function when its input parameters have not changed (other than unnecessary processing). Therefore, an implementation can choose to reduce the number of flags by combining any vertically adjacent states. For example, the changeAggregationLinks flag and UPDATE_PORT_VECTOR state could be eliminated by invoking the updateConversationPortVector() function in the UPDATE_MASK state just before invoking the updateConversationMasks() function, removing the conditional setting of the changeAggregationLinks flag from the updateConversationPortVector() function, and changing any other place that sets changeAggregationLinks to set changeCSCD instead. Ultimately, this type of consolidation could result in a single flag causing an exit from IDLE to a single state that executes all of the functions in sequence and returns to IDLE.

## 6.7 Configuration capabilities and restrictions

### 6.7.1 Use of system and port priorities

The Link Aggregation Group identifier (LAG ID) format specified by this standard cannot represent two or more LAGs that share the same combination of {SK, TL}. The use of static keys in combination with size restriction (e.g., for a System with six Aggregation Ports that have the same operational Key, restricting the size of any aggregation to four Aggregation Ports or fewer) leads to one LAG.

In Systems that have limited aggregation capability of this form, the following algorithm shall be used to determine the subset of Aggregation Ports that can be aggregated together:

a) The System Aggregation Priority of each System is an eight-octet binary number, formed by using the Actor_Oper_System_Priority as the two most significant octets and the Actor_Oper_System as the least significant six octets. For a given Actor and Partner, the System with the numerically lower value of System Aggregation Priority has the higher priority.

b) The Port Aggregation Priority of each Aggregation Port is a four-octet binary number, formed by using the Actor_Port_Priority as the two most significant octets and the Port Number as the two least significant octets. For any given set of Aggregation Ports, the Aggregation Port with the numerically lower value of Port Aggregation Priority has the higher priority.

c) Aggregation Ports shall be selected for aggregation by each System based upon the Port Aggregation Priority assigned by the System with the higher System Aggregation Priority, starting with the highest priority Aggregation Port of the System with the higher priority, and working downward through the ordered list of Port Aggregation Priority values for the N Aggregation Ports, applying the particular constraints imposed on the System concerned.

d) For each link that a given System cannot include in the aggregation, the Selection Logic identifies the Selection state of the corresponding Aggregation Port as STANDBY, preventing the link from becoming active. The synchronization state signaled in transmitted LACPDUs for such links is FALSE.

e) The selection algorithm is reapplied upon changes in the membership of the LAG (for example, if a link fails, or if a new link joins the group) and any consequent changes to the set of active links are made accordingly.

The use of the standby capability is discussed in Annex C.

### 6.7.2 Dynamic allocation of operational Keys

In some circumstances, the use of System and port priorities might prove to be insufficient to generate the optimum aggregation among the set of links connecting a pair of Systems. A System can have a limited aggregation capability that cannot be simply expressed as a limit on the total number of links in the aggregation. The full description of its restrictions might be that it can aggregate together only particular subsets of links, and the sizes of the subsets need not all be the same.

NOTE 1—An example would be an implementation organized such that, for a set of four links A through D, it would be possible to operate with {A+B+C+D} as a single aggregation, or operate with {A+B} and {C+D} as two separate aggregations, or operate as four Solitary links; however, all other aggregation possibilities (such as {A+C} and {B+D}) would not be achievable by the implementation.

In such circumstances, it is permissible for the System with the higher System Aggregation Priority (i.e., the numerically lower value) to dynamically modify the operational Key value associated with one or more of the Aggregation Ports; the System with the lower priority shall not attempt to modify operational Key values for this purpose. Operational Key changes made by the higher priority System maintain its highest priority Aggregation Port in the aggregate as an active link. Successive operational Key changes, if they occur, progressively reduce the number of Aggregation Ports in the aggregation. The original operational Key value is maintained for the highest priority Aggregation Port in the aggregate.

NOTE 2—Restricting operational Key changes in the manner described prevents the case where both Partner Systems involved have limited capability and both attempt to make operational Key changes; this could be a non-converging process, as a change by one participant can cause the other participant to make a change, which in turn causes the first participant to make a change—and so on, ad infinitum.

This approach effectively gives the higher priority System permission to search the set of possible configurations in order to find the best combination of links given its own and its Partner's configuration constraints. The reaction of the Partner System to these changes can be determined by observing the changes in the synchronization state of each link. A System performing operational Key changes should allow at least 4 s for the Partner System to change its synchronization state from FALSE to TRUE.

In the course of normal operation an Aggregation Port can dynamically change its operating characteristics (e.g., data rate, point-to-point operation). It is permissible (and appropriate) for the operational Key value associated with such an Aggregation Port to change with the corresponding changes in the operating characteristics of the link so that the operational Key value always correctly reflects the aggregation capability of the link. Operational Key changes that reflect such dynamic changes in the operating characteristics of a link can be made by either System without restriction.

### 6.7.3 Link Aggregation on shared-medium links

The Link Aggregation Control Protocol cannot detect the presence of multiple Aggregation-aware devices on the same link. Hence, shared-medium links shall be treated as Solitary, with transmission/reception of LACPDUs disabled on such Aggregation Ports.

### 6.7.4 Selection Logic variants

Two variants of the Selection Logic rules are described in this subclause:

a) The first accommodates implementations that operate in a manner that minimizes disturbance of existing aggregates at the expense of the deterministic characteristics of the logic described in 6.4.12.2.

b) The second accommodates implementations that limit the number of Aggregators that are available for use to fewer than the number of Aggregation Ports supported.

#### 6.7.4.1 Reduced reconfiguration

By removing the constraint that the Aggregator chosen is always the Aggregator associated with the Aggregation Port having the lowest Actor_Port_Number in the set of Aggregation Ports in an aggregation, an implementation can minimize the degree to which changes in the membership of a given aggregation result in changes of connectivity at higher layers. As there would still be the same number of Aggregators and Aggregation Ports with a given operational Key value, any Aggregation Port will still always be able to find an appropriate Aggregator to which to attach itself; however, the configuration achieved over time (i.e., after a series of link disconnections, reconnections, or reconfigurations) with this relaxed set of rules would not necessarily be the same as the configuration achieved if all Systems involved were reset, given the rules stated in 6.4.12.2.

#### 6.7.4.2 Limited Aggregator availability

By removing the constraint that there are always as many Aggregators as Aggregation Ports, an implementation can limit the number of Aggregator Client interfaces available to higher layers while maintaining the ability for each Aggregator to serve multiple Aggregation Ports. This has the same effect as removing the assumption that Aggregators and their associated Aggregation Ports have the same operational Key value. An Aggregator can be explicitly disabled by setting the Aggregator_Enabled variable to FALSE; as a result, the MAC_Enabled parameter of the ISS between the Aggregator and the Aggregator Client would have a value of FALSE.

In this scenario, any Aggregation Port(s) that cannot find a suitable Aggregator to which to attach themselves will simply wait in the DETACHED state until an Aggregator becomes available, with a synchronization state of FALSE.

### 6.7.5 LACP configuration for dual-homed Systems

A Link Aggregation System is said to be dual-homed (or multi-homed) when it has a set of Aggregation Ports intended to be connected to two (or more) Partner Systems with the restriction that only one LAG is formed from that set of Aggregation Ports. This restriction is achieved by configuring the Administrative Keys and assigning the Operational Keys so that the following conditions are true:

a) All Aggregation Ports in the set are allocated the same Operational Key value, and that value is different from the Operational Key value allocated to any Aggregation Ports that are not in the set; and

b) Only one Aggregator has an Operational Key value that is the same as the Operational Key value of the Aggregation Ports in the set.

The result is that only one Aggregator is available for the set of Aggregation Ports.

Figure 6-24 shows two examples of dual-homed systems. In Figure 6-24(a), the dual-homed System is connected to two different Partner Systems. In this case, the two Aggregation Ports cannot select the same

Aggregator, and only one Aggregator is available; therefore, the Selection Logic causes only one Aggregation Port of the set to be selected. The result is a single Link Aggregation Group that includes only one link and connects to only one of the Partner Systems. The Aggregation Port that is not initially selected is held in reserve. If the link attached to the initially selected Aggregation Port fails, the other Aggregation Port is selected for a new Link Aggregation Group that again includes only one link but connects to the other Partner System.

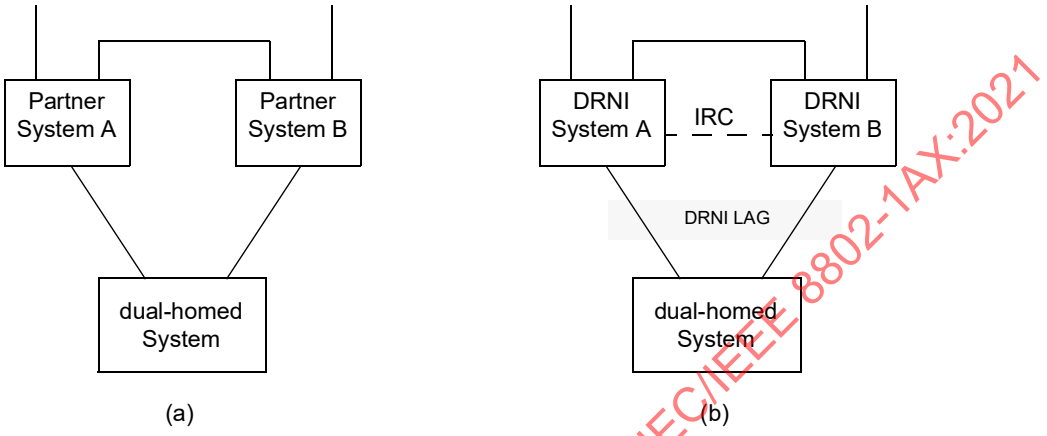



**Figure 6-24—Dual-homed System examples**

In Figure 6-24(b), the dual-homed System is connected to two different DRNI Systems in a DRNI (9.1). The normal DRNI operation allows both DRNI Systems to present themselves as a single Partner System with the same System Identifier and Key values on both links; as a result, both links become active in a single LAG. If the Intra-Relay Connection (IRC) fails, however, the two DRNI Systems use different System Identifier values. This situation prevents both links from forming a single LAG. In this case, the restriction that the dual-homed system has only a single Aggregator available for these Aggregation Ports causes one Aggregation Port to be SELECTED and the other to be UNSELECTED.

In both cases, the recommended default configuration of 6.4.14.2 would cause both links to become active in separate LAGs, with the resulting possibility of forming a data loop. Using the restricted configuration for a dual-homed System guarantees that no loop is formed without relying on any external loop prevention protocol.

# 7. Management

## 7.1 Overview

This clause provides the layer management specification for Link Aggregation. It provides the objects, attributes, and behaviors to support Link Aggregation.

The layout of this clause takes the same form as Clause 30 of IEEE Std 802.3-2018. It identifies a common management model and framework applicable to the IEEE 802.1AX managed elements, identifies those elements, and defines their managed objects, attributes, and behaviors in a protocol-independent language.

It defines facilities comprised of a set of statistics and actions needed to provide IEEE 802.1AX management services. The Procedural Model provides a formal description of the relationship between Link Aggregation and the layer management facilities.

This management specification has been developed in accordance with the OSI management architecture as specified in the ISO Management Framework document, ISO/IEC 7498-4:1989. It is independent of any particular management application or management protocol.

The management facilities defined in this standard can be accessed both locally and remotely. Thus, the layer management specification provides facilities that can be accessed from within a station or can be accessed remotely by means of a peer-management protocol operating between application entities.

In this standard, no peer management facilities are necessary for initiating or terminating normal protocol operations or for handling abnormal protocol conditions. Since these activities are subsumed by the normal operation of the protocol, they are not considered to be a function of layer management and are, therefore, not discussed in this clause.

Implementation of part or all of layer management is not a requirement for conformance to any other clause of this standard.

The improper use of some of the facilities described in this clause can cause serious disruption of the network. In accordance with ISO management architecture, any necessary security provisions are provided by the agent in the Local System Environment. This can be in the form of specific security features or in the form of security features provided by the peer communication facilities.

### 7.1.1 Systems management overview

Within the ISO Open Systems Interconnection (OSI) architecture, the need to handle the special problems of initializing, terminating, and monitoring ongoing activities and assisting in their operations, as well as handling abnormal conditions, is recognized. These needs are collectively addressed by the systems management component of the OSI architecture.

A management protocol is required for the exchange of information between systems on a network. This management standard is independent of any particular management protocol.

This management standard, in conjunction with the management standards of other layers, provides the means to perform various management functions. IEEE 802.1AX management collects information needed from, and provides a means to exercise control over, Link Aggregation.

### 7.1.2 Management model

This standard describes management of Link Aggregation in terms of a general model of management of resources within the open systems environment. The model, which is described in ISO/IEC 10040:1992, is briefly summarized here.

Management is viewed as a distributed application modeled as a set of interacting management processes. These processes are executed by systems within the open environment. A managing system executes a managing process that invokes management operations. A managed system executes a process that is receptive to these management operations and provides an interface to the resources to be managed. A managed object is the abstraction of a resource that represents its properties as seen by (and for the purpose of) management. Managed objects respond to a defined set of management operations. Managed objects are also capable of emitting a defined set of notifications.

A managed object is a management view of a resource. The resource can be a logical construct, function, physical device, or anything subject to management. Managed objects are defined in terms of four types of elements:

a) *Attributes.* Data-like properties (as seen by management) of a managed object.
b) *Actions.* Operations that a managing process can perform on an object or its attributes.
c) *Notifications.* Unsolicited reports of events that can be generated by an object.
d) *Behavior.* The way in which managed objects, attributes, and actions interact with the actual resources they model and with each other.

The preceding items are defined in 7.3 of this standard in terms of the template requirements of ISO/IEC 10165-4:1992.

Some of the functions and resources within IEEE 802.1AX devices are appropriate targets for management. They have been identified by specifying managed objects that provide a management view of the functions or resources. Within this general model, the IEEE 802.1AX device is viewed as a managed device. It performs functions as defined by the applicable standard for such a device. Managed objects providing a view of those functions and resources appropriate to the management of the device are specified. The purpose of this standard is to define the object classes associated with the devices in terms of their attributes, operations, notifications, and behavior.

## 7.2 Managed objects

### 7.2.1 Introduction

This clause identifies the Managed Object classes for IEEE 802.1AX components within a managed system. It also identifies which managed objects and packages are applicable to which components.

All counters defined in this specification are assumed to be wraparound counters. Wraparound counters are those that automatically go from their maximum value (or final value) to zero and continue to operate. These unsigned counters do not provide for any explicit means to return them to their minimum (zero), i.e., reset. Because of their nature, wraparound counters need to be read frequently enough to avoid loss of information. When a counter has a maximum increment rate specified at one speed of operation, and that counter is appropriate to a higher speed of operation, then the maximum increment rate at that higher speed of operation is as shown in Equation (7–1).

$$\text{Maximum increment rate specified} \times \left( \frac{\text{higher speed of operation in Mb/s}}{\text{specified speed of operation in Mb/s}} \right) \quad (7\text{–}1)$$

unless otherwise indicated.

### 7.2.2 Overview of managed objects

Managed objects provide a means to

— Identify a resource
— Control a resource
— Monitor a resource

#### 7.2.2.1 Text description of managed objects

In case of conflict, the formal behavior definitions in  take precedence over the text descriptions in this subclause.

**oAggPortDebugInformation**
A single instance of oAggPortDebugInformation can be contained within oAggregationPort. This managed object class provides optional additional information that can assist with debugging and fault finding in Systems that support Link Aggregation.

**oAggPortStats**
A single instance of oAggPortStats can be contained within oAggregationPort. This managed object class provides optional additional statistics related to LACP and Marker protocol activity on an instance of an Aggregation Port that is involved in Link Aggregation.

**oAggregationPort**
oAggregationPort is contained within oAggregator. An instance of this managed object class is present for each Aggregation Port that is part of the aggregation represented by the oAggregator instance. This managed object class provides the basic management controls necessary to allow an instance of an Aggregation Port to be managed, for the purposes of Link Aggregation.

**oAggregator**
oAggregator is contained within oDRNI. Since oDRNI is optional, oAggregator can be the top-most managed object class of the Link Aggregation containment tree. The oAggregator managed object class provides the management controls necessary to allow an instance of an Aggregator to be managed.

**oDRNI**
oDRNI is optional, and when present, is the top-most managed object class of the tree shown in Figure 7-1. Note that this managed object class (or oAggregator, if oDRNI is not present) can be contained within another superior managed object class. Such containment is expected, but is outside the scope of this international standard. The oDRNI managed object class provides the management controls necessary to allow an instance of a DRNI Gateway to be managed.

### 7.2.3 Containment

A containment relationship is a structuring relationship for managed objects in which the existence of a managed object is dependent on the existence of a containing managed object. The contained managed object is said to be the subordinate managed object, and the containing managed object the superior managed object. The containment relationship is used for naming managed objects. The local containment relationships among object classes are depicted in the entity relationship diagram Figure 7-1. The figure shows the names of the object classes and whether a particular containment relationship is one-to-one or one-to-many. For further requirements on this topic, see IEEE Std 802.1F™-1993 [B2].

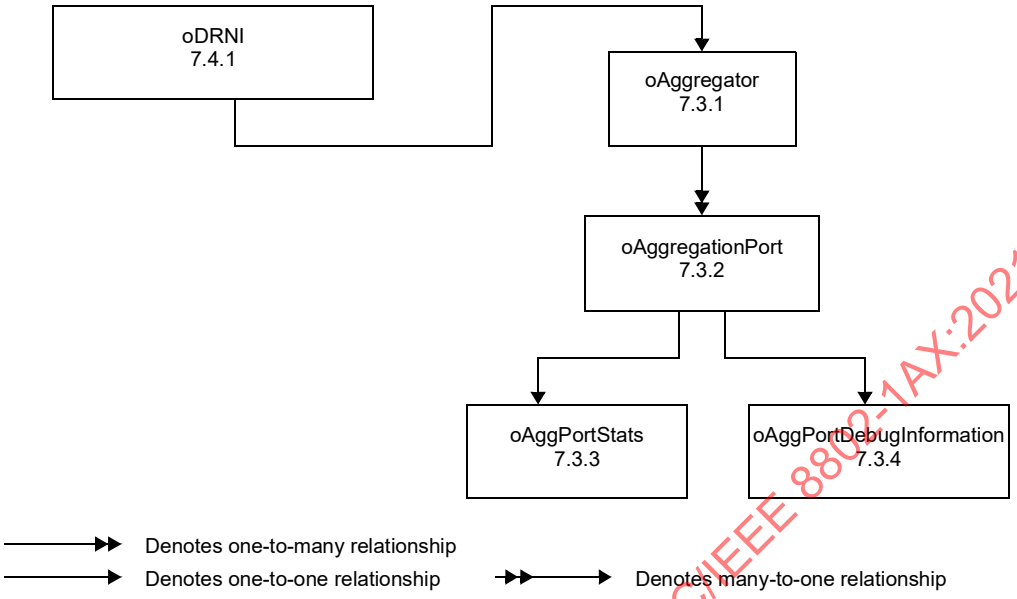



**Figure 7-1—Link aggregation entity relationship diagram**

### 7.2.4 Naming

The name of an individual managed object is hierarchically defined within a managed system. For example, an Aggregator might be identified as "aggregator 3, Aggregation Port 13," that is, Aggregation Port 13 of aggregator 3 within the managed system.

### 7.2.5 Capabilities

This standard makes use of the concept of *packages* as defined in ISO/IEC 10165-4:1992 as a means of grouping behavior, attributes, actions, and notifications within a managed object class definition. Packages can be either mandatory or conditional, i.e., present if a given condition is true. Within this standard *capabilities* are defined, each of which corresponds to a set of packages, which are components of a number of managed object class definitions and which share the same condition for presence. Implementation of the Basic and Mandatory packages is the minimum requirement for claiming conformance to IEEE 802.1AX Management. Implementation of an entire optional capability is required in order to claim conformance to that capability, unless stated otherwise. The capabilities and packages for IEEE 802.1AX Management are specified in Table 7–1.

**Table 7–1—Link Aggregation capabilities**

| | Object name | Object type | Operations supported | Basic package (mandatory) | Mandatory package (mandatory) | Recommended package (optional) | Optional package (optional) | Aggregation Port statistics (optional) | Aggregation Port debug information (optional) | Per-Service Frame Distribution package (optional) | DRNI package (optional) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| oAggregator (7.3.1) | | | | | | | | | | | |
| | aAggID | ATTRIBUTE | GET | X | | | | | | | |
| | aAggDescription | ATTRIBUTE | GET | | X | | | | | | |
| | aAggName | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggActorSystemID | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggActorSystemPriority | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggAggregateOrIndividual | ATTRIBUTE | GET | | X | | | | | | |
| | aAggActorAdminKey | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggActorOperKey | ATTRIBUTE | GET | | X | | | | | | |
| | aAggMACAddress | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPartnerSystemID | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPartnerSystemPriority | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPartnerOperKey | ATTRIBUTE | GET | | X | | | | | | |
| | aAggAdminState | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggOperState | ATTRIBUTE | GET | | X | | | | | | |
| | aAggTimeOfLastOperChange | ATTRIBUTE | GET | | X | | | | | | |
| | aAggDataRate | ATTRIBUTE | GET | | X | | | | | | |
| | aAggOctetsTxOK | ATTRIBUTE | GET | | | X | | | | | |
| | aAggOctetsRxOK | ATTRIBUTE | GET | | | X | | | | | |
| | aAggFramesTxOK | ATTRIBUTE | GET | | X | | | | | | |
| | aAggFramesRxOK | ATTRIBUTE | GET | | X | | | | | | |
| | aAggMulticastFramesTxOK | ATTRIBUTE | GET | | | | X | | | | |
| | aAggMulticastFramesRxOK | ATTRIBUTE | GET | | | | X | | | | |
| | aAggBroadcastFramesTxOK | ATTRIBUTE | GET | | | | X | | | | |
| | aAggBroadcastFramesRxOK | ATTRIBUTE | GET | | | | X | | | | |
| | aAggFramesDiscardedOnTx | ATTRIBUTE | GET | | | X | | | | | |
| | aAggFramesDiscardedOnRx | ATTRIBUTE | GET | | | X | | | | | |
| | aAggFramesWithTxErrors | ATTRIBUTE | GET | | | X | | | | | |
| | aAggFramesWithRxErrors | ATTRIBUTE | GET | | | X | | | | | |
| | aAggUnknownProtocolFrames | ATTRIBUTE | GET | | | X | | | | | |
| | aAggLinkUpDownNotificationEnable | ATTRIBUTE | GET-SET | | X | | | | | | |
| | nAggLinkUpNotification | NOTIFICATION | | | X | | | | | | |
| | nAggLinkDownNotification | NOTIFICATION | | | X | | | | | | |
| | aAggPortList | ATTRIBUTE | GET | | | X | | | | | |

**Table 7–1—Link Aggregation capabilities *(continued)***

| Object name | Object type | Operations supported | Basic package (mandatory) | Mandatory package (mandatory) | Recommended package (optional) | Optional package (optional) | Aggregation Port statistics (optional) | Aggregation Port debug information (optional) | Per-Service Frame Distribution package (optional) | DRNI package (optional) |
|---|---|---|---|---|---|---|---|---|---|---|
| aAggCollectorMaxDelay | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortAlgorithm | ATTRIBUTE | GET-SET | | | | | | | X | |
| aAggAdminConvServiceMap | ATTRIBUTE | GET-SET | | | | | | | X | |
| aAggAdminConvLinkMap | ATTRIBUTE | GET-SET | | | | | | | X | |
| aAggAdminDiscardWrongConversation | ATTRIBUTE | GET-SET | | | | | | | X | |
| aAggOperDiscardWrongConversation | ATTRIBUTE | GET | | | | | | | X | |
| aAggConvLinkDigest | ATTRIBUTE | GET | | | | | | | X | |
| aAggConvServiceDigest | ATTRIBUTE | GET | | | | | | | X | |
| aAggPartnerPortAlgorithm | ATTRIBUTE | GET | | | | | | | X | |
| aAggPartnerConvServiceDigest | ATTRIBUTE | GET | | | | | | | X | |
| aAggPartnerConvLinkDigest | ATTRIBUTE | GET | | | | | | | X | |
| aAggActiveLinks | ATTRIBUTE | GET | | | | | | | X | |
| oAggregationPort (7.3.2) | | | | | | | | | | |
| aAggPortID | ATTRIBUTE | GET | X | | | | | | | |
| aAggPortActorSystemPriority | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortActorSystemID | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortActorAdminKey | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortActorOperKey | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortPartnerAdminSystemPriority | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortPartnerOperSystemPriority | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortPartnerAdminSystemID | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortPartnerOperSystemID | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortPartnerAdminKey | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortPartnerOperKey | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortSelectedAggID | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortAttachedAggID | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortActorPort | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortActorPortPriority | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortPartnerAdminPort | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortPartnerOperPort | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortPartnerAdminPortPriority | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortPartnerOperPortPriority | ATTRIBUTE | GET | | X | | | | | | |
| aAggPortActorAdminState | ATTRIBUTE | GET-SET | | X | | | | | | |
| aAggPortActorOperState | ATTRIBUTE | GET | | X | | | | | | |

**Table 7–1—Link Aggregation capabilities *(continued)***

| | Object name | Object type | Operations supported | Basic package (mandatory) | Mandatory package (mandatory) | Recommended package (optional) | Optional package (optional) | Aggregation Port statistics (optional) | Aggregation Port debug information (optional) | Per-Service Frame Distribution package (optional) | DRNI package (optional) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | aAggPortPartnerAdminState | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggPortPartnerOperState | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPortAggregateOrIndividual | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPortOperConversationPasses | ATTRIBUTE | GET | | | | | | | X | |
| | aAggPortOperConversationCollected | ATTRIBUTE | GET | | | | | | | X | |
| | aAggPortAdminLinkNumber | ATTRIBUTE | GET-SET | | | | | | | X | |
| | aAggPortOperLinkNumber | ATTRIBUTE | GET | | | | | | | X | |
| | aAggPortPartnerLinkNumber | ATTRIBUTE | GET | | | | | | | X | |
| | aAggPortWtrTime | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggPortWtrRevertive | ATTRIBUTE | GET-SET | | X | | | | | | |
| | aAggPortWtrWaiting | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPortActorLacpVersion | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPortPartnerLacpVersion | ATTRIBUTE | GET | | X | | | | | | |
| | aAggPortProtocolDA | ATTRIBUTES | GET-SET | | X | | | | | | |
| oAggPortStats (7.3.3) | | | | | | | | | | | |
| | aAggPortStatsID | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsLACPDUsRx | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsMarkerPDUsRx | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsMarkerResponsePDUsRx | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsUnknownRx | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsIllegalRx | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsLACPDUsTx | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsMarkerPDUsTx | ATTRIBUTE | GET | | | | | X | | | |
| | aAggPortStatsMarkerResponsePDUsTx | ATTRIBUTE | GET | | | | | X | | | |
| oAggPortDebugInformation (7.3.4) | | | | | | | | | | | |
| | aAggPortDebugInformationID | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugRxState | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugLastRxTime | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugMuxState | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugMuxReason | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugActorSyncTransitionCount | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugPartnerSyncTransitionCount | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugActorChangeCount | ATTRIBUTE | GET | | | | | | X | | |
| | aAggPortDebugPartnerChangeCount | ATTRIBUTE | GET | | | | | | X | | |

**Table 7–1—Link Aggregation capabilities *(continued)***

| | Object name | Object type | Operations supported | Basic package (mandatory) | Mandatory package (mandatory) | Recommended package (optional) | Optional package (optional) | Aggregation Port statistics (optional) | Aggregation Port debug information (optional) | Per-Service Frame Distribution package (optional) | DRNI package (optional) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| oDRNI (7.4.1) | | | | | | | | | | | |
| | aDrniID | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniDescription | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniName | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniAggregator | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniAggregationPortList | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeGatewayAlgorithm | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeAggregatorConversationPasses | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniHomeGatewayConversationPasses | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniGatewayAdminState | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniGatewayOperState | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniGatewayTimeOfLastOperChange | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniProtocolDA | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniAggregatorSystem | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniAggregatorSystemPriority | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniAggregatorKey | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeCscdGatewayControl | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeDRClientGatewayControl | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeGatewayEnableMask | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeGatewayPreferenceMask | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeAdminGatewayConvServiceMap | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeGatewayConvServiceDigest | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniHomeGatewayAvailableMask | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniIntraRelayPort | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeAdminIrpState | ATTRIBUTE | GET-SET | | | | | | | | X |
| | aDrniHomeOperIrpState | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborOperIrpState | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborAggregatorConversationPasses | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborGatewayConversationPasses | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborSystem | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborSystemPriority | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborDrniKey | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniSequenceNumbers | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborAggregatorAlgorithm | ATTRIBUTE | GET | | | | | | | | X |

**Table 7–1—Link Aggregation capabilities *(continued)***

| | Object name | Object type | Operations supported | Basic package (mandatory) | Mandatory package (mandatory) | Recommended package (optional) | Optional package (optional) | Aggregation Port statistics (optional) | Aggregation Port debug information (optional) | Per-Service Frame Distribution package (optional) | DRNI package (optional) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | aDrniNeighborAggregatorConvServiceDigest | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborAggregatorConvLinkDigest | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborPartnerSystemPriority | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborPartnerSystem | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborPartnerAggregatorKey | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborCscdState | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborActiveLinks | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborGatewayAlgorithm | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborGatewayConvServiceDigest | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborGatewayAvailableMask | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniNeighborGatewayPreferenceMask | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniDRCPDUsRx | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniIllegalRx | ATTRIBUTE | GET | | | | | | | | X |
| | aDrniDRCPDUsTx | ATTRIBUTE | GET | | | | | | | | X |
| Common Attributes Template | | | | | | | | | | | |
| | aCMCounter | ATTRIBUTE | GET | | X | X | X | X | X | | |



## 7.3 Management for Link Aggregation

### 7.3.1 Aggregator managed object class

This subclause formally defines the behaviors for the oAggregator managed object class, attributes, and notifications.

Some of the attributes that are part of the definition of the oAggregator managed object class are derived by summing counter values from attributes of other objects; e.g., to generate a count of received frames for the Aggregator, the individual value for each Aggregation Port contributes to the sum. Where calculations of this form are used, the values that contribute to the Aggregator's attributes are *increments* in the values of the component attributes, not their absolute values. As any Aggregation Port is potentially only temporarily attached to its current Aggregator, the count values it contributes to the Aggregator's counters are the increments in its values that it has experienced during the period of time that it has been attached to that Aggregator.

The counter values defined for the Aggregator have been formulated as far as possible to make the Aggregator behave like an individual IEEE 802 MAC. The counts of frames received and transmitted are formulated to reflect the counts that would be expected by the Aggregator Client; they do not include frames transmitted and received as part of the operation of LACP or the Marker protocol, only frames that pass through the interface between the Aggregator Client and the Aggregator. However, as LACP and the Marker protocol are, as far as the individual MACs are concerned, part of their Aggregator Client, the RX/TX counters for the individual MACs reflect both control and data traffic. As counts of errors at the Aggregation Port level cannot always be cleanly delineated between those that occurred as a result of aggregation activity and those that did not, no attempt has been made to separate these aspects of the Aggregation Port error counts. Therefore, there is not necessarily a direct correspondence between the individual MAC counters and the corresponding derived counters at the Aggregator level.

The counters defined for the Aggregator include values that can apply only to half-duplex links. This is consistent with the approach taken in Link Aggregation that a link that can operate only as a solitary link is nonetheless considered as being attached to an Aggregator. This simplifies the modeling of managed objects for links that can operate in either half or full duplex and ensures a consistent presentation of the attributes regardless of the type of links attached to the Aggregator.

NOTE—The operation of Auto-Negotiation can mean that a given link can operate in full duplex or half duplex, depending upon the capabilities of the device(s) connected to it. Keeping the management view the same regardless of a link's current mode of operation allows a consistent management approach to be taken across all types of links.

#### 7.3.1.1 Aggregator attributes

##### 7.3.1.1.1 aAggID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The unique identifier allocated to this Aggregator by the local System. This attribute identifies an Aggregator instance among the subordinate managed objects of the containing object. This value is read-only.

NOTE—The aAggID is represented in the SMIv2 MIB as an ifIndex; see D.4.1.

##### 7.3.1.1.2 aAggDescription

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOR DEFINED AS:

A human-readable text string containing information about the Aggregator. This string could include information about the distribution algorithm in use on this Aggregator; for example, "Aggregator 1, Dist Alg=Dest MAC address." This string is read-only. The contents are vendor specific.

##### 7.3.1.1.3 aAggName

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOR DEFINED AS:

A human-readable text string containing a locally significant name for the Aggregator. This string is read-write.

#### 7.3.1.1.4 aAggActorSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:

MACAddress

BEHAVIOR DEFINED AS:

A 6-octet read-write MAC address value used as a unique identifier for the System that contains this Aggregator.

#### 7.3.1.1.5 aAggActorSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

A 2-octet read-write value indicating the priority value associated with the Actor's System Identifier.

#### 7.3.1.1.6 aAggAggregateOrIndividual

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOR DEFINED AS:

A read-only Boolean value that, when TRUE, indicates the Aggregator can represent an LAG consisting of multiple Aggregation Ports. When FALSE, it indicates that the Aggregator can support only a Solitary link because the Aggregation Port that has selected the Aggregator has a value of FALSE for the Aggregation bit in either aAggPortActorOperState or aAggPortPartnerOperState.

#### 7.3.1.1.7 aAggActorAdminKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:

The current administrative value of the Key for the Aggregator. This value is read-write.

#### 7.3.1.1.8 aAggActorOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:

The current operational value of the Key for the Aggregator. The administrative Key value can differ from the operational Key value for the reasons discussed in 6.7.2. This is a 16-bit

read-only value. The meaning of particular Key values is of local significance. This value is read-only.

#### 7.3.1.1.9 aAggMACAddress

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOR DEFINED AS:
A 6-octet read-only value carrying the individual MAC address assigned to the Aggregator.

#### 7.3.1.1.10 aAggPartnerSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOR DEFINED AS:
A 6-octet read-only MAC address value consisting of the unique identifier for the current protocol Partner of this Aggregator.

#### 7.3.1.1.11 aAggPartnerSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:
A 2-octet read-only value that indicates the priority value associated with the Partner's System Identifier.

#### 7.3.1.1.12 aAggPartnerOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:
The current operational value of the Key for the Aggregator's current protocol Partner. This is a 16-bit read-only value.

#### 7.3.1.1.13 aAggAdminState

ATTRIBUTE

APPROPRIATE SYNTAX:
An ENUMERATED VALUE that has one of the following entries:
up
down

BEHAVIOR DEFINED AS:
This read-write value defines the administrative state of the Aggregator. A value of "up" indicates that the operational state of the Aggregator (aAggOperState) is permitted to be either up or down. A value of "down" forces the operational state of the Aggregator to be down. Changes to the administrative state affect the operational state of the Aggregator only,

not the operational state of any Aggregation Ports. A GET operation returns the current administrative state. A SET operation changes the administrative state to a new value.

#### 7.3.1.1.14 aAggOperState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
up
down

BEHAVIOR DEFINED AS:

This read-only value defines the operational state of the Aggregator. An operational state of "up" indicates that the Aggregator is available for use by the Aggregator Client; a value of "down" indicates that the Aggregator is not available for use by the Aggregator Client.

#### 7.3.1.1.15 aAggTimeOfLastOperChange

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The time at which the interface entered its current operational state, in terms of centiseconds since the system was last reset. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a value of zero. The ifLastChange object in the Interfaces MIB defined in IETF RFC 2863 is a suitable object for supplying a value for aAggTimeOfLastOperChange. This value is read-only.

#### 7.3.1.1.16 aAggDataRate

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The current data rate, in bits per second, of the aggregate link. The value is calculated as the sum of the data rate of each link in the aggregation. This attribute is read-only.

#### 7.3.1.1.17 aAggOctetsTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 1 230 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the data and padding octets transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation. The count does not include octets transmitted by the Aggregator in frames that carry LACPDUs or Marker PDUs (7.3.3.1.7, 7.3.3.1.8, and 7.3.3.1.9) or frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

#### 7.3.1.1.18 aAggOctetsRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 1 230 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the data and padding octets received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation. The count does not include octets received in frames that carry LACP or Marker PDUs (7.3.3.1.2, 7.3.3.1.3, and 7.3.3.1.4) or frames discarded by the Frame Collection function of the Aggregator (7.3.1.1.26). This value is read-only.

#### 7.3.1.1.19 aAggFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the data frames transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames transmitted by the Aggregator that carry LACP or Marker PDUs (7.3.3.1.7, 7.3.3.1.8, and 7.3.3.1.9) or frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

#### 7.3.1.1.20 aAggFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the data frames received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames that carry LACP or Marker PDUs (7.3.3.1.2, 7.3.3.1.3, and 7.3.3.1.4) or frames discarded by the Frame Collection or Aggregator Parser/Multiplexer functions of the Aggregator (7.3.1.1.26, 7.3.1.1.28, and 7.3.1.1.29). This value is read-only.

#### 7.3.1.1.21 aAggMulticastFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the data frames transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation, to a group DA other than the broadcast address. The count does not include frames transmitted by the Aggregator that carry LACP or Marker PDUs (7.3.3.1.7, 7.3.3.1.8, and 7.3.3.1.9) or frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

#### 7.3.1.1.22 aAggMulticastFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the data frames received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation, that were addressed to an active group address other than the broadcast address. The count does not include frames that carry LACP or Marker PDUs (7.3.3.1.2, 7.3.3.1.3, and 7.3.3.1.4) or frames discarded by the Frame Collection or Aggregator Parser/Multiplexer functions of the Aggregator (7.3.1.1.26, 7.3.1.1.28, and 7.3.1.1.29). This value is read-only.

#### 7.3.1.1.23 aAggBroadcastFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the broadcast data frames transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames transmitted by the Aggregator that carry LACP or Marker PDUs (7.3.3.1.7, 7.3.3.1.8, and 7.3.3.1.9) or frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

#### 7.3.1.1.24 aAggBroadcastFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of the broadcast data frames received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames that carry LACP or Marker PDUs (7.3.3.1.2, 7.3.3.1.3, and 7.3.3.1.4) or frames discarded by the Frame Collection or Aggregator Parser/Multiplexer functions of the Aggregator (7.3.1.1.26, 7.3.1.1.28, and 7.3.1.1.29). This value is read-only.

#### 7.3.1.1.25 aAggFramesDiscardedOnTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of data frames requested to be transmitted by this Aggregator that were discarded by the Frame Distribution function of the Aggregator when conversations are reallocated to different Aggregation Ports, due to the requirement to ensure that the conversations are flushed on the old Aggregation Ports in order to maintain proper frame ordering (B.3), or

discarded as a result of excessive collisions by Aggregation Ports that are (or have been) members of the aggregation. This value is read-only.

#### 7.3.1.1.26 aAggFramesDiscardedOnRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of data frames, received on all Aggregation Ports that are (or have been) members of the aggregation, that were discarded by the Frame Collection function of the Aggregator as they were received on Aggregation Ports whose Frame Collection function was disabled. This value is read-only.

#### 7.3.1.1.27 aAggFramesWithTxErrors

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of data frames requested to be transmitted by this Aggregator that experienced transmission errors within the Aggregator. This value is read-only.

#### 7.3.1.1.28 aAggFramesWithRxErrors

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of data frames discarded on reception by all Aggregation Ports that are (or have been) members of the aggregation, or that were discarded by the Frame Collection function of the Aggregator, or that were discarded by the Aggregator due to the detection of an illegal Slow Protocols PDU (7.3.3.1.6). This value is read-only.

#### 7.3.1.1.29 aAggUnknownProtocolFrames

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOR DEFINED AS:

A count of data frames discarded on reception by all Aggregation Ports that are (or have been) members of the aggregation, due to the detection of an unknown Slow Protocols PDU (7.3.3.1.5). This value is read-only.

#### 7.3.1.1.30 aAggPortList

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF INTEGERs that match the syntax of aAggPortID.

BEHAVIOR DEFINED AS:

The value of this read-only attribute contains the list of Aggregation Ports that are currently selected or attached to the Aggregator (6.3.9, 6.3.14, and 6.4.13). An empty list indicates that there are no Aggregation Ports attached. Each integer value in the list carries an aAggPortID attribute value (7.3.2.1.1).

#### 7.3.1.1.31 aAggLinkUpDownNotificationEnable

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
enabled
disabled

BEHAVIOR DEFINED AS:

When set to “enabled,” Link Up and Link Down notifications are enabled for this Aggregator. When set to “disabled,” Link Up and Link Down notifications are disabled for this Aggregator. This value is read-write.

#### 7.3.1.1.32 aAggCollectorMaxDelay

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The value of this 16-bit read-write attribute defines the maximum delay, in tens of microseconds, that can be imposed by the Frame Collector between receiving a frame from an Aggregator Parser, and either delivering the frame to its Aggregator Client or discarding the frame (see 6.2.3.1.1).

#### 7.3.1.1.33 aAggPortAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of a 3-octet OUI or CID and one following octet.

BEHAVIOR DEFINED AS:

This read-write value identifies the algorithm used by the Aggregator to assign frames to a Port Conversation ID. Table 8-1 provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings.

#### 7.3.1.1.34 aAggAdminConvLinkMap

ATTRIBUTE

APPROPRIATE SYNTAX:

An array of SEQUENCE OF INTEGERs that match the syntax of aAggPortOperLinkNumber.

BEHAVIOR DEFINED AS:

There are 4096 read-write entries in the aAggAdminConvLinkMap, indexed by Port Conversation ID. Each contains administrative values of the link selection preference list for the referenced Port Conversation ID. This selection preference list is a sequence of aAggPortAdminLinkNumbers for each Port Conversation ID, in the order of preference, highest to lowest, for the corresponding link to carry that Port Conversation ID. A 16-bit zero value is used to indicate that no link is assigned to carry the associated Port Conversation ID.

#### 7.3.1.1.35 aAggAdminDiscardWrongConversation

ATTRIBUTE

APPROPRIATE SYNTAX:

```
INTEGER {
    FORCE_TRUE (1),
    FORCE_FALSE (2),
    AUTO (3)
}
```

BEHAVIOR DEFINED AS:

The administrative value that determines whether the Aggregator discards a frame that is received from an Aggregation Port with a Port Conversation ID that has a value of FALSE in the Port_Oper_Conversation_Mask. The value "FORCE_TRUE" indicates that such frames are to be discarded, the value "FORCE_FALSE" indicates that such frames are to be forwarded, and the value "AUTO" indicates that such frames are to be discarded only when the actor and partner agree on the algorithms and mapping tables used to map frames to Aggregation Ports. Its value is set to "FORCE_FALSE" by default. Support of the "AUTO" and "FORCE_TRUE" values are optional (5.3.2). This value is read-write.

#### 7.3.1.1.36 aAggAdminConvServiceMap

ATTRIBUTE

APPROPRIATE SYNTAX:

An array of SEQUENCE OF INTEGERs that match the syntax of Service IDs (8.2).

BEHAVIOR DEFINED AS:

There are 4096 read-write entries in the aAggAdminConvServiceMap, indexed by Port Conversation ID. Each contains a set of zero or more Service IDs (8.2), unique within the array, that map to that Port Conversation ID. Frames with Service IDs not contained in the map are not mapped to any Port Conversation ID and are discarded.

#### 7.3.1.1.37 aAggOperDiscardWrongConversation

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOR DEFINED AS:

The operational value that determines whether the Aggregator discards a frame that is received from an Aggregation Port with a Port Conversation ID that has a value of FALSE in the Port_Oper_Conversation_Mask. This value is read-only.

#### 7.3.1.1.38 aAggConvLinkDigest

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The value for the digest of aAggAdminConvLinkMap (7.3.1.1.34). This value is read-only.

#### 7.3.1.1.39 aAggConvServiceDigest

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The value for the digest of aAggAdminConvServiceMap (7.3.1.1.36). The value is NULL when the distribution algorithm specified by aAggPortAlgorithm (7.3.1.1.33) does not use the aAggAdminConvServiceMap. This value is read-only.

#### 7.3.1.1.40 aAggPartnerPortAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of a 3-octet OUI or CID and one following octet.

BEHAVIOR DEFINED AS:

The operational value of the distribution algorithm in use by the LACP Partner. This value is read-only.

#### 7.3.1.1.41 aAggPartnerConvLinkDigest

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The operational value of the digest of the aAggAdminConvLinkMap (7.3.1.1.34) in use by the LACP Partner. This value is read-only.

#### 7.3.1.1.42 aAggPartnerConvServiceDigest

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The operational value of the digest of aAggAdminConvServiceMap (7.3.1.1.36) in use by the LACP Partner. This value is read-only.

#### 7.3.1.1.43 aAggActiveLinks

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF INTEGERs that match the syntax of aAggPortOperLinkNumber.

BEHAVIOR DEFINED AS:

The value of this read-only attribute contains the list of the operational Link_Numbers of Aggregation Ports that are currently active (i.e., collecting) on the Aggregator. An empty list indicates that there are no Aggregation Ports active. Each integer value in the list carries an aAggPortOperLinkNumber attribute value (7.3.2.1.28).

#### 7.3.1.2 Aggregator Notifications

##### 7.3.1.2.1 nAggLinkUpNotification

NOTIFICATION

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

When aAggLinkUpDownNotificationEnable is set to “enabled,” a Link Up notification is generated when the Operational State of the Aggregator changes from “down” to “up.” When aAggLinkUpDownNotificationEnable is set to “disabled,” no Link Up notifications are generated. The notification carries the identifier of the Aggregator whose state has changed.

##### 7.3.1.2.2 nAggLinkDownNotification

NOTIFICATION

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

When aAggLinkUpDownNotificationEnable is set to “enabled,” a Link Down notification is generated when the Operational State of the Aggregator changes from “up” to “down.” When aAggLinkUpDownNotificationEnable is set to “disabled,” no Link Down notifications are generated. The notification carries the identifier of the Aggregator whose state has changed.

### 7.3.2 Aggregation Port managed object class

This subclause formally defines the behaviors for the oAggregationPort managed object class attributes.

#### 7.3.2.1 Aggregation Port Attributes

##### 7.3.2.1.1 aAggPortID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The unique identifier allocated to this Aggregation Port by the local System. This attribute identifies an Aggregation Port instance among the subordinate managed objects of the containing object. This value is read-only.

NOTE—The aAggPortID is represented in the SMIv2 MIB as an ifIndex; see D.4.1.

#### 7.3.2.1.2 aAggPortActorSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:
A 2-octet read-write value used to define the priority value associated with the Actor's System Identifier.

#### 7.3.2.1.3 aAggPortActorSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOR DEFINED AS:
A 6-octet read-write MAC address value that defines the value of the System Identifier for the System that contains this Aggregation Port.

#### 7.3.2.1.4 aAggPortActorAdminKey

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:
The current administrative value of the Key for the Aggregation Port. This is a 16-bit read-write value. The meaning of particular Key values is of local significance.

#### 7.3.2.1.5 aAggPortActorOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:
The current operational value of the Key for the Aggregation Port. This is a 16-bit read-only value. The meaning of particular Key values is of local significance.

#### 7.3.2.1.6 aAggPortPartnerAdminSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:
A 2-octet read-write value used to define the administrative value of priority associated with the Partner's System Identifier. The assigned value is used, along with the value of aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey, aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.

#### 7.3.2.1.7 aAggPortPartnerOperSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:
A 2-octet read-only value indicating the operational value of priority associated with the Partner's System Identifier. The value of this attribute contains the manually configured value carried in aAggPortPartnerAdminSystemPriority when there is no protocol Partner.

#### 7.3.2.1.8 aAggPortPartnerAdminSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOR DEFINED AS:
A 6-octet read-write MACAddress value representing the administrative value of the Aggregation Port's protocol Partner's System Identifier. The assigned value is used, along with the value of aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminKey, aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.

#### 7.3.2.1.9 aAggPortPartnerOperSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOR DEFINED AS:
A 6-octet read-only MACAddress value representing the current value of the Aggregation Port's protocol Partner's System Identifier. The value of this attribute contains the manually configured value carried in aAggPortPartnerAdminSystemID when there is no protocol Partner.

#### 7.3.2.1.10 aAggPortPartnerAdminKey

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:
The current administrative value of the Key for the protocol Partner. This is a 16-bit read-write value. The assigned value is used, along with the value of aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID, aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.

#### 7.3.2.1.11 aAggPortPartnerOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The current operational value of the Key for the protocol Partner. The value of this attribute contains the manually configured value carried in aAggPortPartnerAdminKey when there is no protocol Partner. This is a 16-bit read-only value.

#### 7.3.2.1.12 aAggPortSelectedAggID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The identifier value of the Aggregator that this Aggregation Port has currently selected. Zero indicates that the Aggregation Port has not selected an Aggregator, either because it is in the process of detaching from an Aggregator or because there is no suitable Aggregator available for it to select. This value is read-only.

#### 7.3.2.1.13 aAggPortAttachedAggID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The identifier value of the Aggregator to which this Aggregation Port is currently attached. Zero indicates that the Aggregation Port is not currently attached to an Aggregator. This value is read-only.

#### 7.3.2.1.14 aAggPortActorPort

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:

The Port Number locally assigned to the Aggregation Port. The Port Number is communicated in LACPDUs as the Actor_Port. This value is read-only.

#### 7.3.2.1.15 aAggPortActorPortPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The priority value assigned to this Aggregation Port. This 16-bit value is read-write.

#### 7.3.2.1.16 aAggPortPartnerAdminPort

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:

The current administrative value of the Port Number for the protocol Partner. This is a 16-bit read-write value. The assigned value is used, along with the value of

aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.

#### 7.3.2.1.17 aAggPortPartnerOperPort

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The operational Port Number assigned to this Aggregation Port by the Aggregation Port's protocol Partner. The value of this attribute contains the manually configured value carried in aAggPortPartnerAdminPort when there is no protocol Partner. This 16-bit value is read-only.

#### 7.3.2.1.18 aAggPortPartnerAdminPortPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The current administrative value of the Port Priority for the protocol Partner. This is a 16-bit read-write value. The assigned value is used, along with the value of aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey, and aAggPortPartnerAdminPort, in order to achieve manually configured aggregation.

#### 7.3.2.1.19 aAggPortPartnerOperPortPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The priority value assigned to this Aggregation Port by the Partner. The value of this attribute contains the manually configured value carried in aAggPortPartnerAdminPortPriority when there is no protocol Partner. This 16-bit value is read-only.

#### 7.3.2.1.20 aAggPortActorAdminState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:

A string of 8 bits, corresponding to the administrative values of Actor_State (6.4.2.3, Figure 6-9). The first bit corresponds to LACP_Activity (the least significant bit of Actor_State), the second bit corresponds to Short_Timeout, the third bit corresponds to Aggregation, the fourth bit corresponds to Synchronization, the fifth bit corresponds to Collecting, the sixth bit corresponds to Distributing, the seventh bit corresponds to Defaulted, and the eighth bit corresponds to Expired (the most significant bit of Actor_State). These values allow administrative control over the operational values of LACP_Activity, Short_Timeout, and Aggregation. This attribute value is read-write.

#### 7.3.2.1.21 aAggPortActorOperState

ATTRIBUTE

APPROPRIATE SYNTAX:
BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:
A string of 8 bits, corresponding to the current operational values of Actor_State (6.4.2.3, Figure 6-9) as transmitted by the Actor in LACPDUs. The bit allocations are as defined in 7.3.2.1.20. This attribute value is read-only.

#### 7.3.2.1.22 aAggPortPartnerAdminState

ATTRIBUTE

APPROPRIATE SYNTAX:
BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:
A string of 8 bits, corresponding to the current administrative value of Actor_State for the protocol Partner. The bit allocations are as defined in 7.3.2.1.20. This attribute value is read-write. The assigned value is used in order to achieve manually configured aggregation.

#### 7.3.2.1.23 aAggPortPartnerOperState

ATTRIBUTE

APPROPRIATE SYNTAX:
BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:
A string of 8 bits, corresponding to the current values of Actor_State in the most recently received LACPDU transmitted by the protocol Partner. The bit allocations are as defined in 7.3.2.1.20. In the absence of an active protocol Partner, this value reflects the manually configured value aAggPortPartnerAdminState. This attribute value is read-only.

#### 7.3.2.1.24 aAggPortAggregateOrIndividual

ATTRIBUTE

APPROPRIATE SYNTAX:
BOOLEAN

BEHAVIOR DEFINED AS:
A read-only Boolean value that, when TRUE, indicates the Aggregation Port can join a LAG consisting of multiple Aggregation Ports. When FALSE, it indicates that the Aggregation Port can support only a Solitary link because the Aggregation bit in either aAggPortActorOperState or aAggPortPartnerOperState is FALSE.

#### 7.3.2.1.25 aAggPortOperConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX:
BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:
A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that a frame mapping to this Port Conversation ID is distributed to this Aggregation Port, and a 0 indicates that it is not.

#### 7.3.2.1.26 aAggPortOperConversationCollected

ATTRIBUTE

APPROPRIATE SYNTAX:
BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:
A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that a frame mapping to this Port Conversation ID can be collected from this Aggregation Port, and a 0 indicates that it cannot.

#### 7.3.2.1.27 aAggPortAdminLinkNumber

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER, 0 to 65535

BEHAVIOR DEFINED AS:
The Link_Number value for the Aggregation Port, configured by the System's administrator, which is unique among all Aggregation Ports that have the same aAggPortActorSystemPriority, aAggPortActorSystemID, and aAggPortActorAdminKey values, and selected from the set of Link_Numbers in the aAggAdminConvLinkMap of any Aggregator with matching aAggActorSystemPriority, aAggActorSystemID, and aAggActorAdminKey values. This value is read-write.

#### 7.3.2.1.28 aAggPortOperLinkNumber

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER, 0 to 65535

BEHAVIOR DEFINED AS:
The operational Link_Number value for this Aggregation Port that is either the same value as aAggPortAdminLinkNumber or the corresponding value of the LACP partner. This value is read-only.

#### 7.3.2.1.29 aAggPortPartnerLinkNumber

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER, 0 to 65535

BEHAVIOR DEFINED AS:
The last-received value of the Partner_Link_Number, or zero if the Aggregator Port is using default values for the Partner or the Partner LACP Version is 1. This value is read-only.

#### 7.3.2.1.30 aAggPortWtrTime

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOR DEFINED AS:
The wait-to-restore (WTR) period, in seconds, that needs to elapse between an Aggregation Port on a LAG coming up (Port_Operational becoming TRUE) and being permitted to become active (transmitting and receiving frames) on the LAG. This value is read-write.

#### 7.3.2.1.31 aAggPortWtrRevertive

ATTRIBUTE

APPROPRIATE SYNTAX:
BOOLEAN

BEHAVIOR DEFINED AS:
Controls revertive or non-revertive mode of operation. When TRUE, the Aggregation Port can become active as soon as the wait-to-restore timer expires regardless of the state of other links in the LAG. When FALSE, the Aggregation Port cannot become active unless there are no other links that can become active in the LAG. The default value is TRUE. This value is read-write.

#### 7.3.2.1.32 aAggPortWtrWaiting

ATTRIBUTE

APPROPRIATE SYNTAX:
BOOLEAN

BEHAVIOR DEFINED AS:
Indicates the Aggregation Port is inhibited from becoming active for an interval (determined by aAggPortWtrTime) after becoming operational or while non-revertive operation is being enforced by the Selection Logic. This value is read-only.

#### 7.3.2.1.33 aAggPortActorLacpVersion

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOR DEFINED AS:
The version number transmitted in LACPDUs on this Aggregation Port. This value is read-only.

#### 7.3.2.1.34 aAggPortPartnerLacpVersion

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOR DEFINED AS:
The version number in the LACPDU most recently received on this Aggregation Port. This value is read-only.

### 7.3.2.2 Aggregation Port Extension Attributes

#### 7.3.2.2.1 aAggPortProtocolDA

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOR DEFINED AS:
A 6-octet read-write MACAddress value specifying the DA (6.2.10.2) to be used when sending Link Aggregation Control and Marker PDUs on this Aggregation Port,

corresponding to the value of Protocol_DA in 6.2.7.1.2, 6.2.9, and 6.5.4.2.1. The default value shall be the IEEE 802.3 Slow_Protocols_Multicast address.

### 7.3.3 Aggregation Port Statistics managed object class

This subclause formally defines the behaviors for the oAggPortStats managed object class attributes.

#### 7.3.3.1 Aggregation Port Statistics attributes

##### 7.3.3.1.1 aAggPortStatsID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

This read-only attribute identifies an Aggregation Port Statistics object instance among the subordinate managed objects of the containing object. The value allocated to this attribute shall be the same as the containing oAggregationPort managed object.

##### 7.3.3.1.2 aAggPortStatsLACPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of valid LACPDUs received on this Aggregation Port. This value is read-only.

##### 7.3.3.1.3 aAggPortStatsMarkerPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of valid Marker PDUs received on this Aggregation Port. This value is read-only.

##### 7.3.3.1.4 aAggPortStatsMarkerResponsePDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of valid Marker Response PDUs received on this Aggregation Port. This value is read-only.

##### 7.3.3.1.5 aAggPortStatsUnknownRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 50 counts per second.

BEHAVIOR DEFINED AS:

The number of frames received that either

— Carry the Slow Protocols EtherType value (Table 6-3), but contain an unknown PDU, or

— Are addressed to the Slow Protocols group MAC Address (57A.3 of IEEE Std 802.3-2018), but do not carry the Slow Protocols EtherType. This value is read-only.

#### 7.3.3.1.6 aAggPortStatsIllegalRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 50 counts per second.

BEHAVIOR DEFINED AS:

The number of frames received that carry the Slow Protocols EtherType value (Table 6-3), but contain a badly formed PDU or an illegal value of Protocol Subtype (57A.3 of IEEE Std 802.3-2018). This value is read-only.

#### 7.3.3.1.7 aAggPortStatsLACPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of LACPDUs transmitted on this Aggregation Port. This value is read-only.

#### 7.3.3.1.8 aAggPortStatsMarkerPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of Marker PDUs transmitted on this Aggregation Port. This value is read-only.

#### 7.3.3.1.9 aAggPortStatsMarkerResponsePDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of Marker Response PDUs transmitted on this Aggregation Port. This value is read-only.

### 7.3.4 Aggregation Port Debug Information managed object class

This subclause formally defines the behaviors for the oAggPortDebugInformation managed object class attributes.

#### 7.3.4.1 Aggregation Port Debug Information attributes

##### 7.3.4.1.1 aAggPortDebugInformationID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

This read-only attribute identifies an LACP Debug Information object instance among the subordinate managed objects of the containing object. The value allocated to this attribute shall be the same as the containing oAggregationPort managed object.

##### 7.3.4.1.2 aAggPortDebugRxState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
current
expired
defaulted
initialize
lacpDisabled
portDisabled

BEHAVIOR DEFINED AS:

This attribute holds the value “current” if the Receive state machine for the Aggregation Port is in the CURRENT state, “expired” if the Receive state machine is in the EXPIRED state, “defaulted” if the Receive state machine is in the DEFAULTED state, “initialize” if the Receive state machine is in the INITIALIZE state, “lacpDisabled” if the Receive state machine is in the LACP_DISABLED state, or “portDisabled” if the Receive state machine is in the PORT_DISABLED state. This value is read-only.

##### 7.3.4.1.3 aAggPortDebugLastRxTime

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The time at which the last LACPDU was received by this Aggregation Port, in terms of centiseconds since the system was last reset. The ifLastChange object in the Interfaces MIB defined in IETF RFC 2863 is a suitable object for supplying a value for aAggPortDebugLastRxTime. This value is read-only.

NOTE—aAggPortDebugLastRxTime was defined in terms of the aTimeSinceSystemReset variable of Annex F and F.2.1 of IEEE Std 802.3-2018 in earlier versions of this standard. aTimeSinceSystemReset and ifLastChange have the same meaning.

##### 7.3.4.1.4 aAggPortDebugMuxState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
detached
waiting

attached
collecting
distributing
collecting_distributing
attach
attachedWtr

BEHAVIOR DEFINED AS:

This attribute holds the value "detached" if the Mux state machine (6.4.13) for the Aggregation Port is in the DETACHED state, "waiting" if the Mux state machine for the Aggregation Port is in the WAITING state, "attached" if the Mux state machine for the Aggregation Port is in the ATTACHED state, "collecting" if the Mux state machine for the Aggregation Port is in the COLLECTING state, "distributing" if the Mux state machine for the Aggregation Port is in the DISTRIBUTING state, "collecting_distributing" if the Mux state machine for the Aggregation Port is in the COLLECTING_DISTRIBUTING state, "attach" if the Mux state machine for the Aggregation Port is in the ATTACH state, and "attachedWtr" if the Mux state machine for the Aggregation Port is in the ATTACHED_WTR state. This value is read-only.

#### 7.3.4.1.5 aAggPortDebugMuxReason

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOR DEFINED AS:

A human-readable text string indicating the reason for the most recent change of Mux machine state. This value is read-only.

#### 7.3.4.1.6 aAggPortDebugActorSyncTransitionCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

Count of the number of times the Actor's Mux state machine (6.4.13) has changed Actor_Oper_Port_State.Synchronization from FALSE to TRUE. This value is read-only.

#### 7.3.4.1.7 aAggPortDebugPartnerSyncTransitionCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

Count of the number of times the Partner's Mux state machine (6.4.13) has changed its synchronization state from FALSE to TRUE. This value is read-only.

#### 7.3.4.1.8 aAggPortDebugActorChangeCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

Count of the number of times the Actor's perception of the LAG ID for this Aggregation Port has changed. This value is read-only.

#### 7.3.4.1.9 aAggPortDebugPartnerChangeCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

Count of the number of times the Partner's perception of the LAG ID (6.3.6.1) for this Aggregation Port has changed. This value is read-only.

## 7.4 Management for Distributed Resilient Network Interconnect

### 7.4.1 DRNI Managed Object Class

This subclause formally defines the behaviors for the oDRNI managed object class attributes.

#### 7.4.1.1 DRNI Attributes

#### 7.4.1.1.1 aDrniID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The unique identifier allocated to this DRNI Gateway by the local System. This attribute identifies a DRNI Gateway Port instance among the subordinate managed objects of the containing object. This value is read-only.

NOTE—The aDrniID is represented in the SMIv2 MIB as an ifIndex; see D.5.

#### 7.4.1.1.2 aDrniDescription

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOR DEFINED AS:

A human-readable text string containing information about the DRNI Gateway. This string is read-only. The contents are vendor specific.

#### 7.4.1.1.3 aDrniName

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOR DEFINED AS:

A human-readable text string containing a locally significant name for the DRNI Gateway. This string is read-write.

#### 7.4.1.1.4 aDrniAggregator

ATTRIBUTE

APPROPRIATE SYNTAX:

An INTEGER that matches the syntax of an Interface Identifier.

BEHAVIOR DEFINED AS:

Read-write Interface Identifier of the Aggregator Port supporting this DRNI Gateway.

#### 7.4.1.1.5 aDrniAggregationPortList

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF INTEGERs that match the syntax of aAggPortID.

BEHAVIOR DEFINED AS:

The value of this read-write attribute contains the list of Aggregation Ports that are assigned to this DRNI Gateway. Each integer value in the list carries an aAggPortID attribute value (7.3.2.1.1).

#### 7.4.1.1.6 aDrniHomeGatewayAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of an OUI or CID and one following octet.

BEHAVIOR DEFINED AS:

This read-write value identifies the algorithm used by the DRNI Gateway to assign frames to a Gateway Conversation ID. 8.2 provides the IEEE 802.1 OUI (00-80-C2) Gateway Algorithm encodings.

#### 7.4.1.1.7 aDrniHomeAggregatorConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that the Port Conversation ID is allowed to be distributed through this DRNI Gateway's Aggregator, and a 0 indicates that it cannot. aDrniHomeAggregatorConversationPasses is referencing the current value of Home_Aggregator_Mask (9.5.2.2).

#### 7.4.1.1.8 aDrniHomeGatewayConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-only current operational vector of Boolean values, with one value for each possible Gateway Conversation ID. A 1 indicates that the Gateway Conversation ID is allowed to pass through this DRNI Gateway Port, and a 0 indicates that it cannot. aDrniHomeGatewayConversationPasses is referencing the current value of Home_Gateway_Mask (9.5.2.2).

#### 7.4.1.1.9 aDrniGatewayAdminState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
up
down

BEHAVIOR DEFINED AS:

This read-write value defines the administrative state of the DRNI Gateway Port. A value of "up" indicates that the operational state of the Gateway (aDrniGatewayOperState) is permitted to be either "up" or "down." A value of "down" forces the operational state of the DRNI Gateway Port to be "down". A GET operation returns the current administrative state. A SET operation changes the administrative state to a new value.

#### 7.4.1.1.10 aDrniGatewayOperState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
up
down

BEHAVIOR DEFINED AS:

This read-only value defines the operational state of the DRNI Gateway Port. An operational state of "up" indicates that the DRNI Gateway Port is available for use by the Distributed Relay Client; a value of "down" indicates that the DRNI Gateway Port is not available for use by the Distributed Relay Client.

#### 7.4.1.1.11 aDrniGatewayTimeOfLastOperChange

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOR DEFINED AS:

The time at which the interface entered its current operational state, in terms of centiseconds since the system was last reset. If the current state was entered prior to the last reinitialization of the local network management subsystem, then this object contains a value of zero. The ifLastChange object in the Interfaces MIB defined in IETF RFC 2863 is a suitable object for supplying a value for aDrniGatewayTimeOfLastOperChange. This value is read-only.

#### 7.4.1.1.12 aDrniProtocolDA

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF 6 OCTETS that match the syntax of a 48-bit MAC Address

BEHAVIOR DEFINED AS:

A 6-octet read-write MAC Address value specifying the DA to be used when sending DRCPDUs, corresponding to the value of DRNI_DA in 9.6.5. Its value is one of the addresses selected from Table 9-1 and its default shall be the IEEE 802.1 Nearest non-TPMR Bridge group address (01-80-C2-00-00-03).

#### 7.4.1.1.13 aDrniAggregatorSystem

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF 6 OCTETS that match the syntax of a 48-bit MAC Address

BEHAVIOR DEFINED AS:

The Aggregator System value to be used by the Aggregator supporting this DRNI Gateway (and the Aggregation Ports assigned to this DRNI Gateway) when paired with a neighbor DRNI System via the Intra-Relay Connection (IRC). This value is read-write.

#### 7.4.1.1.14 aDrniAggregatorSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The Aggregator System Priority value to be used by the Aggregator supporting this DRNI Gateway (and the Aggregation Ports assigned to this DRNI Gateway) when paired with a neighbor DRNI System via the IRC. This value is read-write.

#### 7.4.1.1.15 aDrniAggregatorKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 1 to 65535

BEHAVIOR DEFINED AS:

The Aggregator Key value to be used by the Aggregator supporting this DRNI Gateway (and the Aggregation Ports assigned to this DRNI Gateway) when paired with a neighbor DRNI System via the IRC. This value is read-write.

#### 7.4.1.1.16 aDrniHomeCscdGatewayControl

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOR DEFINED AS:

A read-write Boolean value that, when TRUE, allows the DRNI Gateway Port selection to be based on the CSCD parameters that control the Aggregator Port selection (see 9.5.3.5).

#### 7.4.1.1.17 aDrniHomeDRClientGatewayControl

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOR DEFINED AS:

A read-write Boolean value that, when TRUE, allows the Distributed Relay Client to determine whether to forward frames through the DRNI Gateway Port (see 9.5.3.5).

#### 7.4.1.1.18 aDrniHomeGatewayEnableMask

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-write vector of Boolean values, with one value for each possible Gateway Conversation ID, configured by administration. A 1 indicates frames associated with that Gateway Conversation ID are allowed to pass through this DRNI Gateway Port, and a 0 indicates that such frames are not allowed to pass.

#### 7.4.1.1.19 aDrniHomeGatewayPreferenceMask

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-write vector of Boolean values, with one value for each possible Gateway Conversation ID, configured by administration. A 1 indicates that the DRNI Gateway Port in this DRNI Gateway is the preferred Gateway when both DRNI Gateways have the Gateway Conversation ID enabled in the aDrniHomeGatewayAvailableMask, and a 0 indicates that it is not.

#### 7.4.1.1.20 aDrniHomeAdminGatewayConvServiceMap

ATTRIBUTE

APPROPRIATE SYNTAX:

An array of SEQUENCE OF INTEGERs, that match the syntax of Service IDs (8.2).

BEHAVIOR DEFINED AS:

There are 4096 read-write entries in the aDrniHomeAdminGatewayConvServiceMap variables, indexed by Gateway Conversation ID. Each contains a set of zero or more Service IDs (8.2), unique within the array that map to that Gateway Conversation ID. Frames with Service IDs not contained in the map are not mapped to any Gateway Conversation ID and are discarded.

#### 7.4.1.1.21 aDrniHomeGatewayConvServiceDigest

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The value for the digest of aDrniHomeAdminGatewayConvServiceMap (7.4.1.1.20). The value is NULL when the distribution algorithm specified by aDrniAggregationPortList (7.4.1.1.5) does not use the aDrniHomeAdminGatewayConvServiceMap. This value is read-only.

#### 7.4.1.1.22 aDrniHomeGatewayAvailableMask

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-only vector of Boolean values, with one value for each possible Gateway Conversation ID. A 1 indicates this DRNI Gateway Port is eligible to be selected to pass that Gateway Conversation ID, and a 0 indicates that it is not eligible. aDrniHomeGatewayAvailableMask is referencing the Gateway_Available_Mask in the Home_Gateway_State variable.

#### 7.4.1.1.23 aDrniIntraRelayPort

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER matching the syntax of an Interface Identifier.

BEHAVIOR DEFINED AS:

Read-write Interface Identifier (ifIndex) of the Port supporting the Intra Relay Port of this DRNI Gateway.

#### 7.4.1.1.24 aDrniHomeAdminIrpState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:

A string of 8 bits, corresponding to the administrative values of IRP_State (9.6.2.3 and Figure 9-13). The first two bits (the least significant bits of IRP_State) are reserved, the third bit corresponds to Short_Timeout, the fourth bit corresponds to Sync, the fifth bit corresponds to IRC_Data, the sixth bit corresponds to DRNI, the seventh bit corresponds to Defaulted, and the eighth bit corresponds to Expired (the most significant bit of IRP_State). These values allow administrative control over the operational values of Short_Timeout and IRC_Data. This attribute value is read-write.

#### 7.4.1.1.25 aDrniHomeOperIrpState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:

A string of 8 bits, corresponding to the current operational value of IRP_State (9.6.2.3 and Figure 9-13) as transmitted in DRCPDUs. The bit allocations are as described for aDrniHomeAdminIrpState (7.4.1.1.24). This attribute value is read-only.

#### 7.4.1.1.26 aDrniNeighborOperIrpState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:

A string of 8 bits, corresponding to the current operational values of IRP_State (9.6.2.3 and Figure 9-13) for the DRCP Neighbor. The bit allocations are as described for aDrniHomeAdminIrpState (7.4.1.1.24). This attribute value is read-only.

#### 7.4.1.1.27 aDrniNeighborAggregatorConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that the Port Conversation ID is to be distributed through the IRP to the Neighbor Aggregator, and a 0 indicates that it is not. aDrniNeighborAggregatorConversationPasses is referencing the current value of Neighbor_Aggregator_Mask (9.5.2.2).

#### 7.4.1.1.28 aDrniNeighborGatewayConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-only current operational vector of Boolean values, with one value for each possible Gateway Conversation ID. A 1 indicates that the Gateway Conversation ID is to be passed through the Neighbor DRNI Gateway Port via the IRP, and a 0 indicates that it is not. aDrniNeighborGatewayConversationPasses is referencing the current value of Neighbor_Gateway_Mask (9.5.2.2).

#### 7.4.1.1.29 aDrniNeighborSystem

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF 6 OCTETS that match the syntax of a 48-bit MAC Address

BEHAVIOR DEFINED AS:

The MAC Address portion of the System Identifier of the Neighbor DRNI System (connected via the Intra-Relay Port). This value is read-only.

#### 7.4.1.1.30 aDrniNeighborSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The priority portion of the System Identifier of the Neighbor DRNI System (connected via the Intra-Relay Port). This value is read-only.

#### 7.4.1.1.31 aDrniNeighborDrniKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The DRNI key value received from the Neighbor DRNI System (connected via the Intra-Relay Port). This value is read-only.

#### 7.4.1.1.32 aDrniSequenceNumbers

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF INTEGERS

BEHAVIOR DEFINED AS:

The current values of all sequence numbers used by DRCP to coordinate the exchange of Aggregator State, Gateway State, and Gateway Preference variables. The sequence consists of the Home Aggregator_Sequence_Number, Home Gateway_Sequence_Number, Home Gateway_Preference_Sequence_Number, Neighbor Aggregator_Sequence_Number, Neighbor Gateway_Sequence_Number, Neighbor Gateway_Preference_Sequence_Number, Reflected_Aggregator_Sequence_Number, Reflected_Gateway_Sequence_Number, and Reflected_Gateway_Preference_Sequence_Number. These values are read-only.

#### 7.4.1.1.33 aDrniNeighborAggregatorAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of an OUI or CID and one following octet.

BEHAVIOR DEFINED AS:

The Port algorithm used by the Neighbor Aggregator to assign frames to Port Conversation IDs. aDrniSequenceNumbers is referencing the Aggregator_Port_Algorithm in the Neighbor_Aggregator_State variable. This value is read-only.

#### 7.4.1.1.34 aDrniNeighborAggregatorConvServiceDigest

ATTRIBUTE

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The digest of the Neighbor Aggregator's Admin_Conv_Service_Map. aDrniNeighborAggregatorConvServiceDigest is referencing the Aggregator_Conv_Service_Digest in the Neighbor_Aggregator_State variable. This value is read-only.

#### 7.4.1.1.35 aDrniNeighborAggregatorConvLinkDigest

ATTRIBUTE

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The digest of the Neighbor Aggregator's Admin_Conv_Link_Map. aDrniNeighborAggregatorConvLinkDigest is referencing the Aggregator_Conv_Link_Digest in the Neighbor_Aggregator_State variable. This value is read-only.

#### 7.4.1.1.36 aDrniNeighborPartnerSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The priority portion of the System Identifier of the Neighbor Aggregator's Partner. aDrniNeighborPartnerSystemPriority is referencing the Partner_System_Priority in the Neighbor_Aggregator_State variable. This value is read-only.

#### 7.4.1.1.37 aDrniNeighborPartnerSystem

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF 6 OCTETS that match the syntax of a 48-bit MAC Address

BEHAVIOR DEFINED AS:

The MAC Address portion of the System Identifier of the Neighbor Aggregator's Partner. aDrniNeighborPartnerSystem is referencing the Partner_System in the Neighbor_Aggregator_State variable. This value is read-only.

#### 7.4.1.1.38 aDrniNeighborPartnerAggregatorKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER: 0 to 65535

BEHAVIOR DEFINED AS:

The operational key value of the Neighbor Aggregator's Partner. aDrniNeighborPartnerAggregatorKey is referencing the Partner_Oper_Aggregator_Key in the Neighbor_Aggregator_State variable. This value is read-only.

#### 7.4.1.1.39 aDrniNeighborCscdState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOR DEFINED AS:

A read-only string of 8 bits, corresponding to the Aggregator_CSCD_State in the Neighbor_Aggregator_State variable. The first three bits (the least significant bits of CSCD_State) are reserved; the fourth bit corresponds to the Neighbor's value for Home_Admin_CSCD_Gateway_Control; the fifth bit corresponds to the Neighbor Aggregator's operational value for Discard_Wrong_Conversation; and the sixth, seventh, and eighth bits correspond to the Neighbor Aggregator's operational value for differConvLinkDigests, differConvServiceDigests, and differPortAlgorithms, respectively, (the most significant bits of CSCD_State).

#### 7.4.1.1.40 aDrniNeighborActiveLinks

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF INTEGERs that match the syntax of aAggPortOperLinkNumber.

BEHAVIOR DEFINED AS:

A list of the operational Link_Numbers of Aggregation Ports that are currently active (i.e., collecting) on the Neighbor's Aggregator. An empty list indicates that there are no Aggregation Ports active. Each integer value in the list carries an aAggPortOperLinkNumber attribute value (7.3.2.1.28). aDrniNeighborActiveLinks is referencing the Active_LAG_Links in the Neighbor_Aggregator_State variable. This value is read-only.

#### 7.4.1.1.41 aDrniNeighborGatewayAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF OCTETS consisting of an OUI or CID and one following octet.

BEHAVIOR DEFINED AS:

The gateway algorithm used by the Neighbor DRNI Gateway to assign frames to Gateway Conversation IDs. aDrniSequenceNumbers is referencing the Gateway_Algorithm in the Neighbor_Gateway_State variable. This value is read-only.

#### 7.4.1.1.42 aDrniNeighborGatewayConvServiceDigest

ATTRIBUTE

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS:

The digest of the Neighbor DRNI Gateway's aDrniHomeAdminGatewayConvServiceMap (7.4.1.1.20). aDrniNeighborGatewayConvServiceDigest is referencing the Gateway_Conv_Service_Digest in the Neighbor_Gateway_State variable. This value is read-only.

#### 7.4.1.1.43 aDrniNeighborGatewayAvailableMask

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-only vector of Boolean values, with one value for each possible Gateway Conversation ID. A 1 indicates that the Neighbor DRNI Gateway Port is eligible to be selected to pass that Gateway Conversation ID, and a 0 indicates that it is not eligible. aDrniNeighborGatewayAvailableMask is referencing the Gateway_Available_Mask in the Neighbor_Gateway_State variable. This value is read-only.

#### 7.4.1.1.44 aDrniNeighborGatewayPreferenceMask

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS:

A read-only vector of Boolean values, with one value for each possible Gateway Conversation ID. A 1 indicates that the DRNI Gateway Port in the Neighbor DRNI Gateway is the preferred Gateway when both DRNI Gateways have the Gateway Conversation ID enabled in the Gateway_Available_Mask, and a 0 indicates that it is not. aDrniNeighborGatewayPreferenceMask is referencing the Gateway_Preference_Mask in the Neighbor_Gateway_Preference variable. This value is read-only.

#### 7.4.1.1.45 aDrniDRCPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of valid DRCPDUs received on this Intra-Relay Port. This value is read-only.

#### 7.4.1.1.46 aDrniIllegalRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 50 counts per second.

BEHAVIOR DEFINED AS:

The number of frames received on this Intra-Relay Port that carry the DRCP EtherType value (9.6.1.4), but contain a badly formed PDU. This value is read-only.

#### 7.4.1.1.47 aDrniDRCPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOR DEFINED AS:

The number of valid DRCPDUs transmitted on this Intra-Relay Port. This value is read-only.

# 8. Distribution algorithms

The CSCD procedures specified in 6.6 provide a structure for administrative control over the distribution algorithm used by a Link Aggregation Sublayer and for communicating the distribution algorithm in use to an LACP Partner. A Distributed Relay operation (9.2) uses a similar structure for distributing frames between the DRNI Gateway Ports. Each algorithm is capable of mapping any frame to a Port Conversation ID (6.6.1 and 9.5.3.4) or a Gateway Conversation ID (9.5.3.5). Some algorithms associate each frame with a Service Identifier, using one or more of the frame fields specified by IEEE Std 802.1Q, and a manageable table that maps each Service Identifier to a conversation identifier.

This clause specifies the following:

a) The identification method for distribution algorithms (8.1)
b) The advantages of Per-Service Frame Distribution (8.2)
c) Standard distribution algorithms (8.2.1 through 8.2.5) and their identifiers (Table 8-1)

## 8.1 Distribution algorithm identification

Each distribution algorithm is identified by a sequence of 4 octets, structured as shown in Figure 8-1. Distribution algorithm identifiers are used by network administrators to select between algorithms and, in Conversation-sensitive LACP and Distributed Resilient Network Interconnect (DRNI) operation, to check whether partners and neighbors are using the same algorithm.

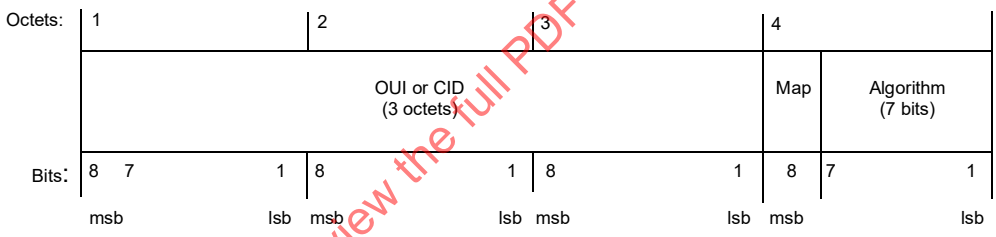



**Figure 8-1—Distribution algorithm identifiers**

The first three octets contain the OUI or CID of the organization defining the distribution algorithm, and the fourth octet contains the value identifying the algorithm assigned by the organization defining the algorithm. The most significant bit of the fourth octet indicates whether the algorithm makes use of a Service ID mapping table (i.e., the Admin_Conv_Service_Map to map Service IDs to Port Conversation IDs, or the Home_Admin_Gateway_Conv_Service_Map to map Service IDs to Gateway Conversation IDs). A value of 1 indicates a map is used; a value of 0 indicates a map is not used. Table 8-1 lists the distribution algorithms specified by this standard and includes identifiers for those algorithms. Use of these standardized algorithms is encouraged, although the inclusion of an OUI or CID in each algorithm identifier supports the definition of proprietary distribution algorithms. The “Unspecified distribution algorithm” identifier 00-80-C2-00 has been reserved for use when the algorithm is unknown (or is not advertised).

**Table 8-1—IEEE per-service distribution algorithms**

| Distribution algorithm | Map | Algorithm identifier |
|---|---|---|
| Unspecified distribution algorithm | No | 00-80-C2-00 |
| Distribution based on C-VIDs (8.2.1) | No | 00-80-C2-01 |
| | Yes | 00-80-C2-81 |
| Distribution based on S-VIDs (8.2.2) | No | 00-80-C2-02 |
| | Yes | 00-80-C2-82 |
| Distribution based on I-SIDs (8.2.3) | No | 00-80-C2-03 |
| | Yes | 00-80-C2-83 |
| Distribution based on TE-SIDs (8.2.4) | No | 00-80-C2-04 |
| | Yes | 00-80-C2-84 |
| Distribution based on Flow Hash (8.2.5) | No | 00-80-C2-05 |
| | Yes | 00-80-C2-85 |
| Reserved | | 00-80-C2-06 to 00-80-C2-80 and 00-80-C2-86 to 00-80-C2-FF |

## 8.2 Per-Service Frame Distribution

Mapping frames to Conversation IDs based on a Service ID can provide the following:

a) **Connectivity Fault Management congruity**—The only thing that Connectivity Fault Management (CFM) (Clause 18 of IEEE Std 802.1Q-2018) PDUs have in common with the data that they protect is, in general, the service ID(s) that they share. Per-Service Frame Distribution ensures that the CFM PDUs traverse the same physical links as their data.

b) **Bidirectional congruity**—Providing a means for the two ends of an Aggregation Group to use the same physical link in both directions for a given service ensures that a failed link or a link experiencing an excessive error rate will affect the fewest possible number of services and in general provide support for protocols that have strict symmetry requirements on their transmit and receive paths, e.g., Precision Time Protocol (PTP) in IEEE Std 1588™ [B3].

c) **Ingress predictability**—Ingress metering is often applied on a per-service basis. Confining a service to a single physical link localizes the meter, facilitating this process.

NOTE—An Aggregation System or DRNI running Per-Service Frame Distribution can interoperate with an Aggregation System or DRNI not running the same Per-Service Frame Distribution. In this case, the Link Aggregation goals listed in 6.1.1 can still be met, but without the advantages of Per-Service Frame Distribution listed in this subclause.

### 8.2.1 Distribution based on C-VLAN Identifier (C-VID)

When distribution is based on the C-VLAN Identifier, the Service ID is the 12-bit VLAN Identifier contained in the C-VLAN tag within the frame (IEEE Std 802.1Q-2018 Figure 9-1). The Service ID of a frame that does not contain a C-VLAN tag in the initial octets of the mac_service_data_unit is defined as 0.

When the Algorithm Identifier indicates that the Service ID mapping table is not used, the Conversation ID is the same value as the Service ID (i.e., the same value as the C-VID).

When distribution based on C-VLAN Identifier is used in a bridge, either the mapping table is used to map the Service ID value of 0 to the PVID value, or the entry in the Admin_Conv_Link_Map for Conversation ID equal to 0 is the same as the entry for Conversation ID equal to the PVID.

### 8.2.2 Distribution based on S-VLAN Identifier (S-VID)

When distribution is based on the S-VLAN Identifier, the Service ID is the 12-bit VLAN Identifier contained in the S-VLAN tag within the frame (IEEE Std 802.1Q-2018 Figure 9-1). The Service ID of a frame that does not contain a S-VLAN tag in the initial octets of the mac_service_data_unit is defined as 0.

When the Algorithm Identifier indicates that the Service ID mapping table is not used, the Conversation ID is the same value as the Service ID (i.e., the same value as the S-VID).

When distribution based on S-VLAN Identifier is used in a bridge, either the mapping table is used to map Service ID value 0 to the PVID value, or the entry in the Admin_Conv_Link_Map for Conversation ID equal to 0 is the same as the entry for Conversation ID equal to the PVID.

### 8.2.3 Distribution based on Backbone Service Instance Identifier (I-SID)

When distribution is based on the Backbone Service Instance Identifier, the Service ID is the 24-bit I-SID contained in the I-Tag within the frame (IEEE Std 802.1Q-2018 Figure 9-2). The Service ID of a frame that does not contain a I-Tag in the initial octets of the mac_service_data_unit is defined as 0.

When the Algorithm Identifier indicates that the Service ID mapping table is not used, the Conversation ID is the same value as the least significant 12 bits of the I-SID.

### 8.2.4 Distribution based on Traffic Engineering Service Instance Identifier (TE-SID)

When distribution is based on the Traffic Engineering Service Instance Identifier, the Service ID is the 32-bit locally assigned identifier representing a series of 3-tuples <ESP_DA, ESP-SA, ESP-VID>, each identifying one of the Ethernet Switched Paths (ESPs) in the Traffic Engineering Service Instance (17.2.10 of IEEE Std 802.1Q-2018).

When the Algorithm Identifier indicates that the Service ID mapping table is not used, the Conversation ID is the same value as the least significant 12 bits of the TE-SID.

NOTE—When Conversation-Sensitive Collection is enabled by setting Admin_Discard_Wrong_Conversation to TRUE or AUTO, all participants in the LAG need to use the same TE-SID values for each TESI; otherwise, significant frame loss can occur.

### 8.2.5 Distribution based on Flow Hash

When distribution is based on the Equal Cost Multi-Path Flow Hash, the Service ID is the 16-bit Flow Hash contained in the Flow Filtering Tag (F-Tag) within the frame (Figure 44-1 of IEEE Std 802.1Q-2018). The Service ID of a frame that does not contain a F-Tag in the initial octets of the mac_service_data_unit is defined as 0.

When the Algorithm Identifier indicates that the Service ID mapping table is not used, the Conversation ID is the same value as the least significant 12 bits of the Service ID (i.e., the same value as the Flow Hash).

# 9. Distributed Resilient Network Interconnect

A Distributed Resilient Network Interconnect (DRNI), comprising two cooperating DRNI Systems, can be used to terminate one end of a Link Aggregation Group (LAG). A system connected to the individual links at the other end uses the LAG, along with the Link Aggregation Control Protocol (LACP), as if those links provide connectivity to a single Aggregation System. DRNIs can thus be deployed independently at either or both ends of any LAG to provide fault tolerance. The two cooperating DRNI Systems provide Distributed Relay functionality to a Distributed Relay Client with a component in each DRNI System.

DRNI operates independently of the (possibly distributed) functionality of the Distributed Relay Client that it supports in each DRNI System. It does not define or constrain ways of distributing arbitrary system functionality so that two systems mimic the behavior of one—achieving such a goal is a difficult task. DRNI can be used to attach a System providing end station functionality to a network, with possible distributed system functions including support for Virtual Router Redundancy Protocol (IETF RFC 5798 [B7]) and File Server applications. Their specification is outside the scope of this standard.

This clause provides an overview of the following:

a) DRNI goals (9.1)
b) Distributed Relay operation (9.2)
c) The Intra-Relay Connection (IRC) connecting the DRNI Systems (9.3)
d) The use of DRNI to protect against link and system failures (9.4)

and specifies the operation of the following in detail:

e) The DRNI Gateway providing Distributed Relay functionality in each DRNI System (9.5)
f) The Distributed Relay Control Protocol (DRCP, 9.6)

Architectural concepts common to this and other IEEE 802.1 standards are used in this clause's specification of DRNI. The reader is encouraged to review Clause 7 of IEEE Std 802.1AC-2016.

The models of operation in this clause provide a basis for specifying the externally observable behavior of the operation and are not intended to place additional constraints on implementations; these can adopt any internal model of operation compatible with the externally observable behavior specified. Conformance of equipment to this standard is purely with respect to observable protocol.

## 9.1 Goals

Link Aggregation (as described in 6.1, 6.2, and 6.3) can aggregate the links that connect two Aggregation Systems (i.e., create a LAG). The Aggregator Clients in each of the Aggregation Systems can use the resulting LAG as if it were a single link, but the LAG is resilient (i.e., provides fault tolerance) to the failure, addition, or removal of individual links in the LAG as long as it provides sufficient bandwidth for the conversations it carries.

DRNI is resilient to system failure while maintaining support for link resilience. Specifically DRNI meets the basic Link Aggregation goals described in 6.1.1 and provides the following:

a) **System redundancy**—An Aggregation System can be replaced by two cooperating DRNI Systems that share access to a DRNI LAG, each attaching to some of the links in that LAG.
b) **System load sharing**—The functionality of the Distributed Relay Clients that use the service provided by a DRNI LAG can be distributed between the DRNI Systems, with conversations directed to one system or the other.

NOTE—A DRNI LAG provides the same Internal Sublayer Service (ISS, IEEE Std 802.1AC) as an individual link. Multiple protocol entities (and potentially different instances of a given protocol entity) can make use of a port providing that service by using standardized protocol discrimination, addressing, and multiplexing functions.

c) **System and link conversation distribution**—Conversation-Sensitive Collection and Distribution (6.6) parameters apply independently to the allocation of conversations to links and to their allocation to each DRNI System's Distributed Relay Client. An Intra-Relay Connection (IRC) allows a conversation associated with a Distributed Relay Client in one DRNI System to be carried by a DRNI LAG link attached to the other DRNI System.

d) **Interoperability**—A DRNI can support a LAG whose links provide connectivity to a single Aggregation System that meets the conformance requirements specified in 5.3 of this standard or specified in prior editions of this standard.

e) **Localized fault recovery**—The effects of DRNI link or system failure can be confined to the DRNI LAG and its attached DRNI Systems using DRCP to provide rapid recovery (see 9.4).

An explicit goal of DRNI is to support the connection of one network to another, where services (each comprising one or more potential conversations) are provisioned across one network, the DRNI LAG, and the second network. Each DRNI System is typically (but not necessarily) a Provider Bridge, as specified in IEEE Std 802.1Q. The use of a DRNI LAG to connect the two networks provides the following:

f) **Administrative independence**—The connected networks can be administered separately and can use different routing and fault recovery protocols.

g) **Distribution independence**—The frame distribution algorithm used to select the DRNI System responsible for relaying frames for a given conversation to and from the attached network can differ from the selection algorithm used by the DRNI attached to the other end of the DRNI LAG to relay frames to and from its own network. Each algorithm's assignment of frames to conversations can also differ.

h) **Inter-network fault isolation**—Failure or recovery of a link or node in one network can be hidden from the other network.

i) **DRNI fault isolation**—The failure or recovery of a link in a DRNI LAG can be hidden from both networks' control protocols. The failure of a DRNI System attached to one network can be hidden from the other network.

Fault and fault recovery scenarios are described in 9.4.

## 9.2 Distributed Relay operation

When a LAG terminates in a single Aggregation System, frames transmitted and received on the LAG are accepted from and delivered to a single Aggregator Client attached to a single Aggregator Port, as shown in Figure 6-3 and described in 6.1, 6.2, and 6.3. When a LAG terminates in a DRNI, frames can be accepted from and delivered to a single Client attached to a single DRNI Gateway Port in one of the DRNI Systems, but can also use Conversation-Sensitive Collection and Distribution (CSCD) to support two DRNI Gateway Ports, one in each DRNI System. The CSCD parameters that allocate conversations to DRNI Gateway Ports are independent of those that allocate conversations to Aggregation Ports of the LAG. An Intra-Relay Connection (IRC) conveys frames passing between a DRNI Gateway Port in one DRNI System and an Aggregation Port in the other and thereby provides Distributed Relay functionality. If sufficient bandwidth is available, DRNI Gateway Port conversation allocation can be independent of the failure of individual links of a DRNI LAG or of a DRNI System or IRC at the other end of the LAG.

Link Aggregation allows the Aggregator Client to treat the LAG as if it were a single link. A DRNI LAG gives the Distributed Relay Client in each of the DRNI Systems access to the same single link. This can require coordination between the Distributed Relay Clients. The allocation of conversations between the

DRNI Gateway Ports can affect the degree of coordination required. For example, allocating all conversations to one of the DRNI Gateway Ports effectively makes the Distributed Relay Client in one of the DRNI Systems "active" and the Distributed Relay Client in the other DRNI System "standby". Allocating some conversations to one of the DRNI Gateway Ports and the remaining conversations to the other creates a "load-sharing" configuration. In some cases the Distributed Relay Clients can each operate independently on their respective set of conversations with little or no coordination, while in other cases the Distributed Relay Clients can operate as a single distributed system function. Any coordination between the Distributed Relay Clients is heavily dependent on the type of functionality provided and is beyond the scope of this standard. Some considerations when the Distributed Relay Client is a VLAN Bridge are discussed in E.1.

Figure 9-1 shows the structure of an example DRNI. DRNI Systems A and B each have five external network connections (MACs a through e, and p through t, respectively). Their immediate client is the Link Aggregation sublayer (operating as specified in Clause 6) in each system. This sublayer aggregates the links the DRNI System uses to participate in the DRNI LAG and provides an Aggregator Port that supports the DRNI System's Distributed Relay sublayer. In the figure these Aggregation Links are attached to MACs d and e in System A and MACs p and q in System B. The Link Aggregation sublayer in each DRNI System can provide additional Aggregator Ports that allow the DRNI System to use other network links (Solitary or aggregated) that are not in the DRNI LAG. In the figure these links could be attached to MACs a through c in System A and MACs r through t in System B.

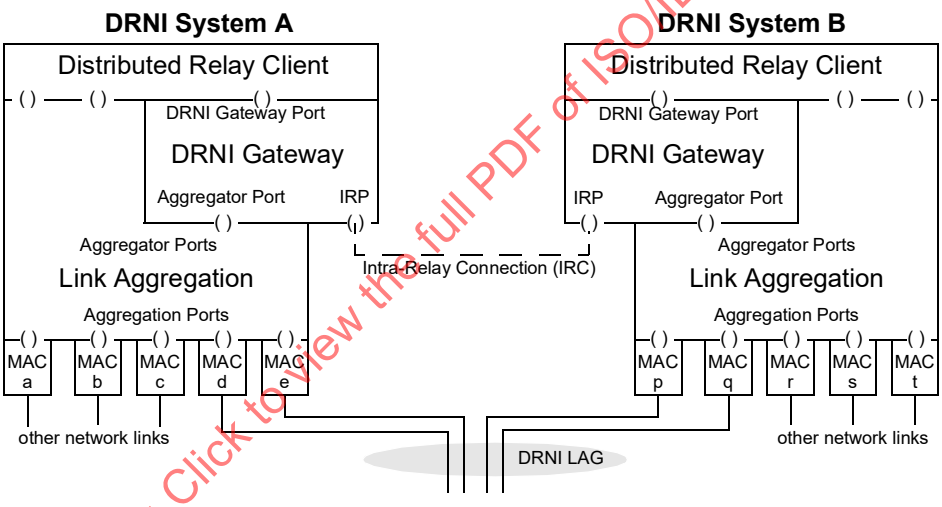



**Figure 9-1—A DRNI**

The Distributed Relay sublayer (DRNI Gateway) uses its Aggregator Port and an Intra-Relay Port (IRP) to support a DRNI Gateway Port that allows the Distributed Relay Client to transmit frames to any of the DRNI LAG links ('Down' frames) and to receive frames from any of the DRNI LAG links ('Up' frames). Frames to or from the Distributed Relay Client for conversations carried by links attached to that DRNI System are relayed between the DRNI Gateway Port and the Aggregator Port. Frames to or from the Distributed Relay Client for conversations carried by links attached to the other DRNI System are relayed between the DRNI Gateway Port and the IRP and are relayed by the other system's DRNI Gateway between its IRP and its Aggregator Port. Figure 9-2 illustrates the relay of frames between the Distributed Relay Client in DRNI System A and an Aggregation Link attached to DRNI System B (left side of figure) and between the Distributed Relay Client in DRNI System B and an Aggregation Link attached to DRNI System A (right side of figure). A DRNI System's DRNI Gateway thus transmits both Up and Down frames, and receives both Up and Down frames, at its IRP.

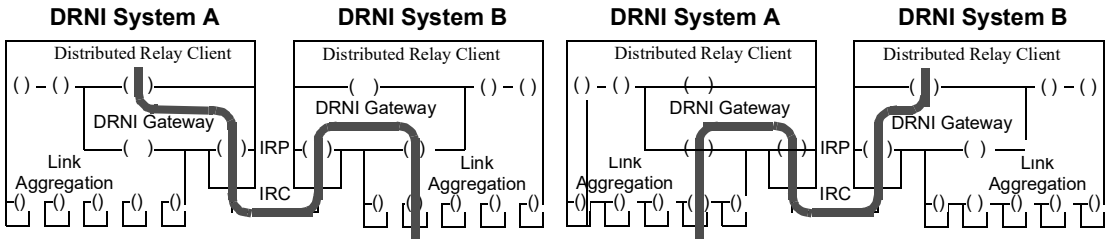



**Figure 9-2—Transmission and reception via the IRC**

The details of providing the IRC are not shown in Figure 9-1 or Figure 9-2; options for implementing the IRC are discussed in 9.3. IRC frames do not carry additional information marking them as Up or Down or associating them with a DRNI Gateway Port or an Aggregation Port, and thus they avoid the need to support additional tagging and detagging operations and frame formats. Instead, DRCP synchronizes each DRNI Gateway's assignment of each conversation to a DRNI Gateway Port (its own or that of its peer DRNI Gateway in the other system). If a system has changed that assignment since synchronization was last achieved, its DRNI Gateway discards frames for that conversation that would otherwise be transmitted to or received from the IRC. Discarding these frames avoids accidentally returning an Up frame received from a DRNI LAG link attached to one DRNI System to another link in the same DRNI LAG in the other DRNI System or forwarding a Down frame from one DRNI Gateway Port to the other.

When IRC connectivity has been established and the DRNI Systems are paired, the LACPDUs transmitted on both DRNI System's Aggregation Ports convey the same Actor System Identifier (comprising the Actor System and Actor System Priority fields as specified in 6.4.2) and Actor Aggregation Key. The values in these fields are determined as described in 9.5.3. A single Aggregation System can thus aggregate links connecting to both DRNI Systems.

A DRNI System can include more than one DRNI Gateway, each with its own Aggregator Port, DRNI Gateway Port, and IRP (providing IRC connectivity to a DRNI Gateway in its companion DRNI System).

## 9.3 Intra-Relay Connection

An IRC is a logical link that supports the ISS at the IRPs of paired DRNI Gateways, one in each DRNI System. DRCP is used to identify and verify Intra-Relay connectivity and guard against physical or logical mis-wiring or administrative error.

Figure 9-3 shows a DRNI that uses a dedicated link for the IRC. A DRNI System that conforms to this standard shall be capable of using a dedicated link implementing an IEEE 802 MAC method, (e.g., as specified by IEEE Std 802.3).

Figure 9-4 shows a DRNI that uses a dedicated LAG for the IRC. In this case the IRP is supported by an Aggregator Port in a Link Aggregation sublayer. The IRC LAG can contain a single Aggregation Link; therefore, a DRNI System supporting an IRP with a Link Aggregation Sublayer meets the requirement to be capable of using a dedicated link for the IRC.

In principle IRC connectivity can be provided by any method capable of supporting the ISS and a separate connectivity association (7.8 of IEEE Std 802.1AC-2016) for each pair of IRPs. For example, Annex E discusses the use of a service instance in Provider Bridged Network (i.e., an S-VLAN) as the IRC between DRNI Gateways in a pair of Provider Edge Bridges (E.2) and the use of a backbone service instance in a Provider Backbone Bridged Network as the IRC between DRNI Gateways in a pair of Backbone Edge Bridges (E.3). Selection and deployment of a particular method ought to take into account the potential for service disruption due to failure or reconfiguration of service supporting components.

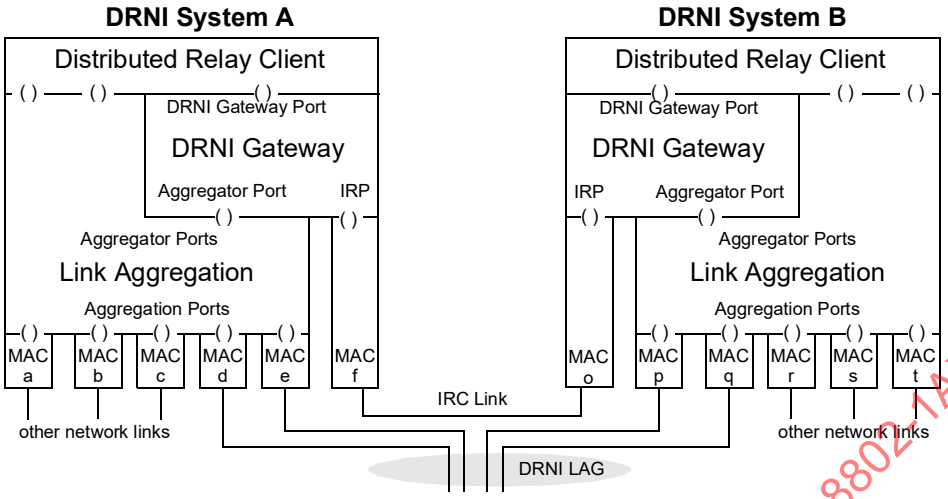



**Figure 9-3—DRNI with dedicated IRC Link**

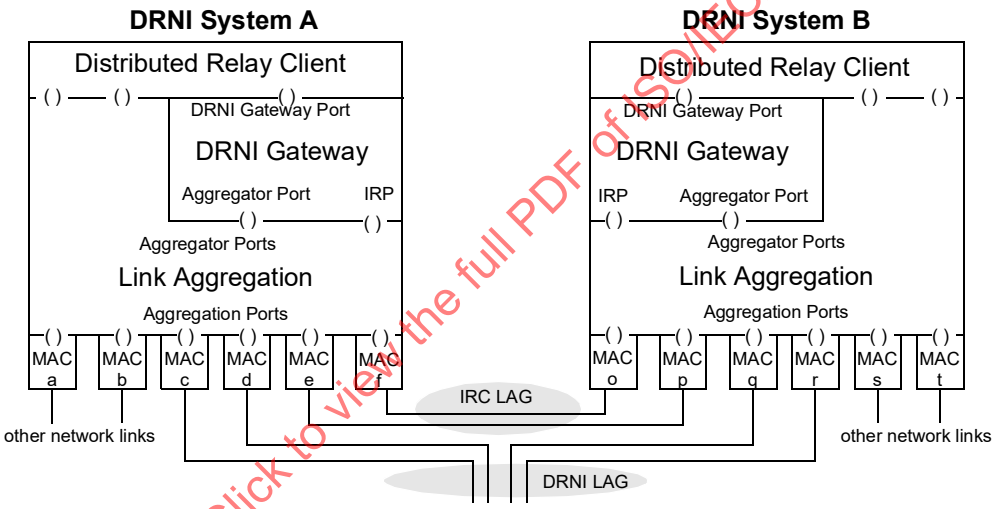



**Figure 9-4—DRNI with dedicated IRC LAG**



## 9.4 Using DRNI

Subclause 9.4 describes how DRNI is used to interconnect networks or systems (9.4.1), how it provides fault recovery (9.4.2), and how it is configured (9.4.3).

### 9.4.1 DRNI connectivity

A DRNI LAG can be used to attach an end station, or a pair of DRNI Systems providing distributed end station functionality, to a network, as shown in Figure 9-5. Distributing the end station functionality between the DRNI Systems is facilitated by the CSCD parameters that allocate conversations to DRNI Systems, but the details of distributing specific end station functionality is beyond the scope of this standard.

Similarly, a DRNI LAG can be used to connect two Bridged Networks, as shown in Figure 9-6. Considerations regarding the use of DRNI in Bridges are discussed in Annex E.

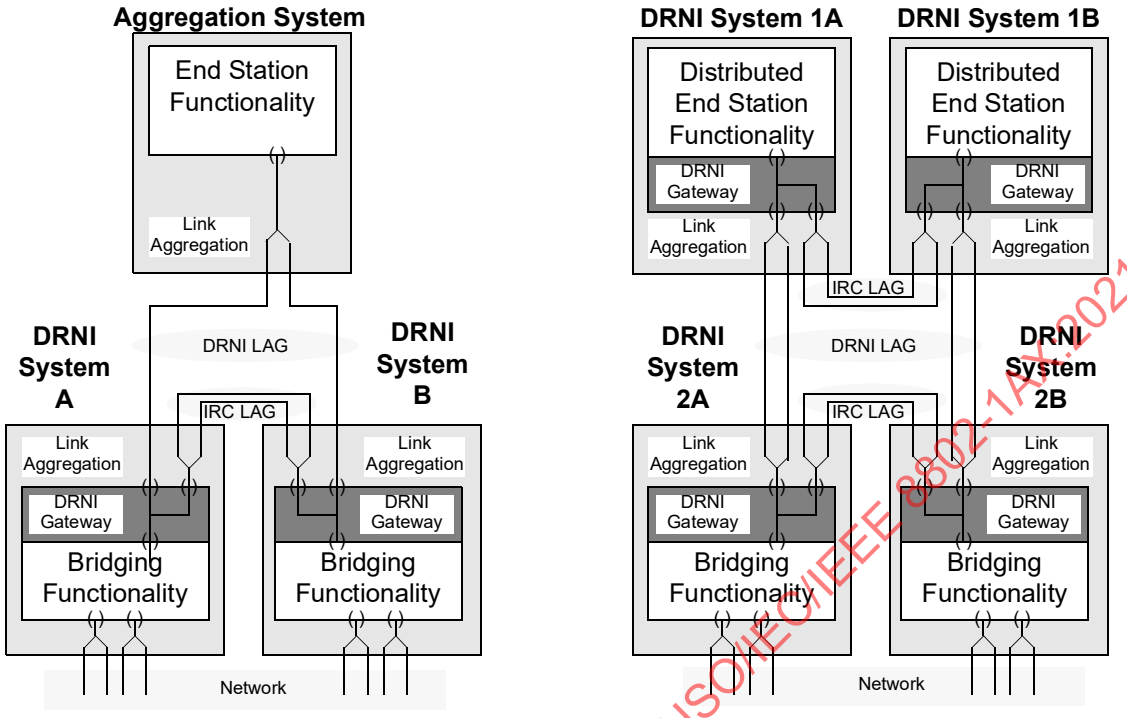



**Figure 9-5—DRNI end station attach**

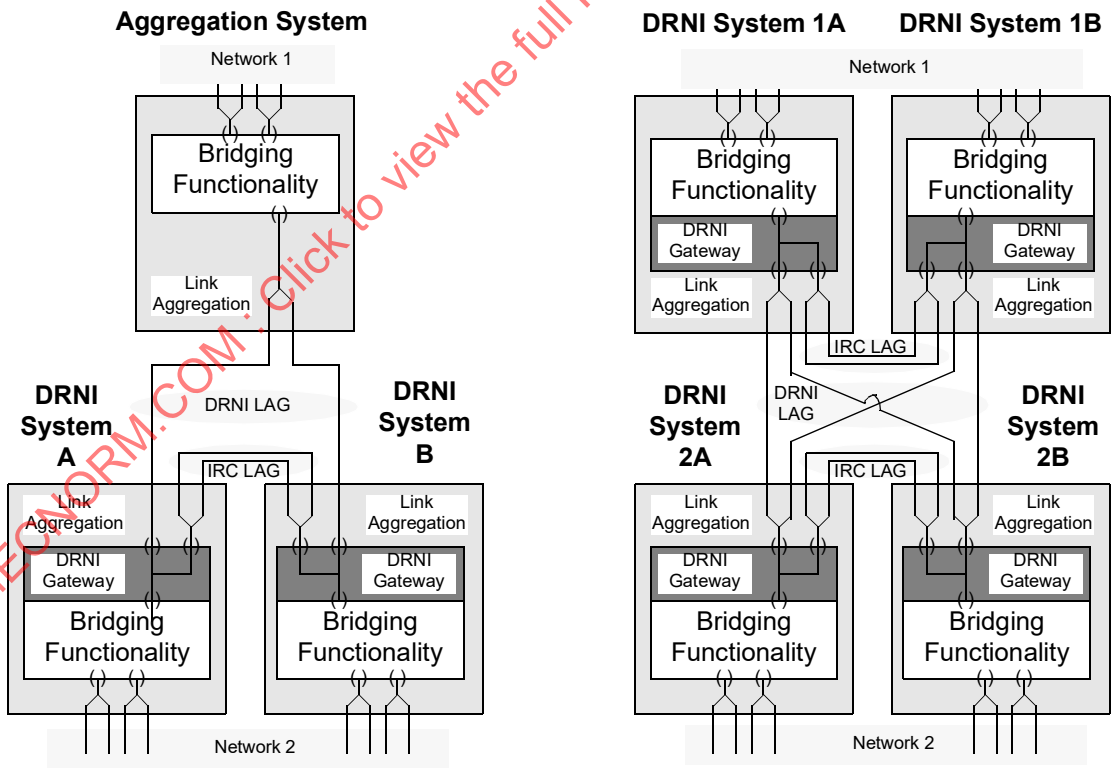



**Figure 9-6—DRNI network attach**

### 9.4.2 DRNI fault recovery

DRNI includes recovery mechanisms for maintaining connectivity across the DRNI LAG in the event of the failure of a link in the DRNI LAG, or a DRNI System, or the IRC.

If an Aggregation Link in the DRNI LAG fails, the conversation(s) allocated to that link are reallocated among the remaining Aggregation Links. If, as a result, a conversation is allocated to a link that terminates on the other DRNI System, the IRC can be used to convey its frames to the original DRNI System so the failure has no impact on the network beyond the DRNI System [9.1e), 9.1i)]. Alternatively the allocation of conversations to DRNI Systems can be configured to be the same as the allocation of conversations to Aggregation Links. This trades isolation of the network from the link failure in favor of reducing the amount of traffic on the IRC.

If the IRC loses connectivity, both DRNI Systems revert to independent operation so each reallocates all conversations to its own Aggregation Links and its own DRNI Gateway Port. When operating independently, each DRNI System advertises its own Home System Identifier rather than the DRNI Aggregator Identifier on its Aggregation Links so the Partner cannot put these links in the same LAG. This results in three scenarios:

a) If the Partner is a DRNI, it can use these links only to support a single LAG and so can use only Aggregation Links terminating on one of the independently operating DRNI Systems.
b) If the Partner is an Aggregation System configured as a dual-homed System (6.7.5), it also can use these links only to support a single LAG and so can use only Aggregation Links terminating on one of the independently operating DRNI Systems.
c) If the Partner is an Aggregation System not configured as a dual-homed System (6.7.5), then it could form a separate LAG to each of the independently operating DRNI Systems.

In the first two scenarios, connectivity is maintained to just one of the DRNI Systems; the resulting situation is analogous to a DRNI System failure. In the third scenario, the resulting behavior depends on the functionality of the Distributed Relay Client supported by the DRNI Systems and on the functionality supported by the Partner Aggregation System. In particular, if all of these Systems are Bridges (e.g., left side of Figure 9-6) and there is connectivity between the DRNI Systems in the Bridged Network, it is possible to create a forwarding loop. Therefore, it is prudent to configure an Aggregation System as a dual-homed System when connected to a DRNI.

The choice of the method used to implement the IRC has a significant impact on its resiliency. A single dedicated IRC Link is a single point of failure. A dedicated IRC LAG provides resiliency. The use of a logical link for the IRC (e.g., as shown in Figure E-2 or Figure E-3) can potentially use any available path through the network and does not fail unless all connectivity is lost between the DRNI Systems.

If a DRNI System fails, the remaining DRNI System reverts to independent operation. Conversations previously allocated to Aggregation Links terminating on the failed DRNI System are reallocated among the Aggregation Links terminating on the remaining DRNI System. Conversations allocated to the DRNI Gateway Port in the failed DRNI System are reallocated to the DRNI Gateway Port on the remaining DRNI System. The failure of a DRNI System on one side of a DRNI LAG looks like the failure of one or more Aggregation Links to the Partner on the other side of the DRNI LAG and thus is hidden from the network beyond the Partner System or DRNI [item h) in 9.1].

The CSCD parameters allow configuring some conversations to be “unprotected”; in other words, they are not reallocated in the event of an Aggregation Link or DRNI System failure.

### 9.4.3 DRNI configuration

Proper operation of DRNI requires coordinated configuration of the DRNI Gateway in each of the two DRNI Systems. The parameters to be configured include the following:

a) DRNI Aggregator Identifier and Key (DRNI_Aggregator_System_Priority, DRNI_Aggregator_System, and DRNI_Aggregator_Key): Determine the operational values for the actor identifier and key in LACPDUs transmitted on the DRNI LAG by the DRNI System to its LACP Partner when the DRNI Gateway is paired with a neighbor. The use and uniqueness requirements are described in 9.5.3.1 and 9.5.3.2.

b) DRNI Gateway Port selection parameters:

   1) Home_Admin_Gateway_Algorithm and, depending on the algorithm, Home_Admin_Gateway_Conv_Service_Map: Configured to be the same in both DRNI Systems. If they differ or have the default value “Unspecified” (Table 8-1), all frames will be forwarded through the DRNI Gateway in the DRNI System with the lowest Home System Identifier (9.5.3.1).
   2) Home_Admin_Gateway_Enable_Mask and Home_Admin_Gateway_Preference_Mask: Can be configured to administratively control which Gateway Conversation IDs are passed at the DRNI Gateway Port on each DRNI System. There are no constraints on how these are configured. With the default values, all frames will be forwarded through the DRNI Gateway in the DRNI System with the lowest Home System Identifier (9.5.3.1). An overview of how these parameters affect the DRNI Gateway Port selection is in 9.5.3.5, and the details are in the DRNI Gateway and Aggregator machine (9.6.12).
   3) Home_Admin_Client_Gateway_Control and Home_Admin_CSCD_Gateway_Control: Allow setting administrative policy over the DRNI Gateway Port selection as described in 9.5.3.5. There are no constraints on how these are configured, although typically they would be the same in both DRNI Systems. Default values are FALSE.

c) The Aggregator supporting the DRNI Gateway and the Aggregation Ports assigned to the DRNI Gateway are indicated by the aDrniAggregator (7.4.1.1.4) and aDrniAggregationPortList (7.4.1.1.5) parameters. The CSCD parameters of the Aggregator and the Aggregation Ports are configured as follows:

   1) Actor_Port_Algorithm, Admin_Conv_Link_Map, Admin_Discard_Wrong_Conversation, and, depending on the algorithm, Admin_Conv_Service_Map: Configured to be the same in both DRNI Systems. If they differ and the Aggregator’s operational Discard_Wrong_Conversation value is TRUE, significant frame loss can result.
   2) The Admin_Link_Number for each Aggregation Port assigned to the DRNI is configured to be unique among the combined set of Aggregation Ports assigned to the DRNI in both DRNI Systems and is in the set of Link_Numbers in the Admin_Conv_Link_Map. If the Admin_Link_Numbers are not unique and the Aggregator’s operational Discard_Wrong_Conversation value is TRUE, significant frame loss can result. If a Admin_Link_Number is not in the set of Link_Numbers in the Admin_Conv_Link_Map, no frames will be distributed to that Aggregation Port.

d) The dedicated link, dedicated Aggregator, or other IRC method supporting the Intra-Relay Port is indicated by the DRNI Gateway’s aDrniIntraRelayPort (7.4.1.1.23) parameter. If this is an Aggregator, the Aggregator and Aggregation Port parameters are configured as follows:

   1) Actor_Admin_System: Unique among the set of Systems with potential links to each other, including the two DRNI Systems.
   2) Actor_Admin_Aggregator_Key: Unique among the Aggregators in this DRNI System. There cannot be any other Aggregators in the DRNI System with the same Actor_Admin_System_Priority, Actor_Admin_System, and Actor_Admin_Aggregator_Key values and Aggregator_Enabled TRUE.

3) The Actor_Admin_Port_System_Priority, Actor_Admin_Port_System, and Actor_Admin_Port_Key for each Aggregation Port assigned to the IRC are configured to be the same as the Actor_Admin_System_Priority, Actor_Admin_System, and Actor_Admin_Aggregator_Key of the Aggregator, respectively. If they differ, the Aggregation Port will not be able to join the IRC LAG.

## 9.5 DRNI Gateway

Figure 9-7 depicts the DRNI Gateway with its primary functional elements and its interface ports. The DRNI Gateway in each DRNI System has three service interfaces:

a) One DRNI Gateway Port supporting the Distributed Relay Client of the DRNI System.

b) One Aggregator Port supported by an Aggregator in a Link Aggregation sublayer with any number of Aggregation Ports configured with the same System Identifier and Key as the Aggregator.

c) One Intra-Relay Port (IRP) supported by an Intra-Relay Connection (IRC) to the Intra-Relay Port of a DRNI Gateway in the paired DRNI System. (Options for implementing the IRC are discussed in 9.3 and Annex E.)

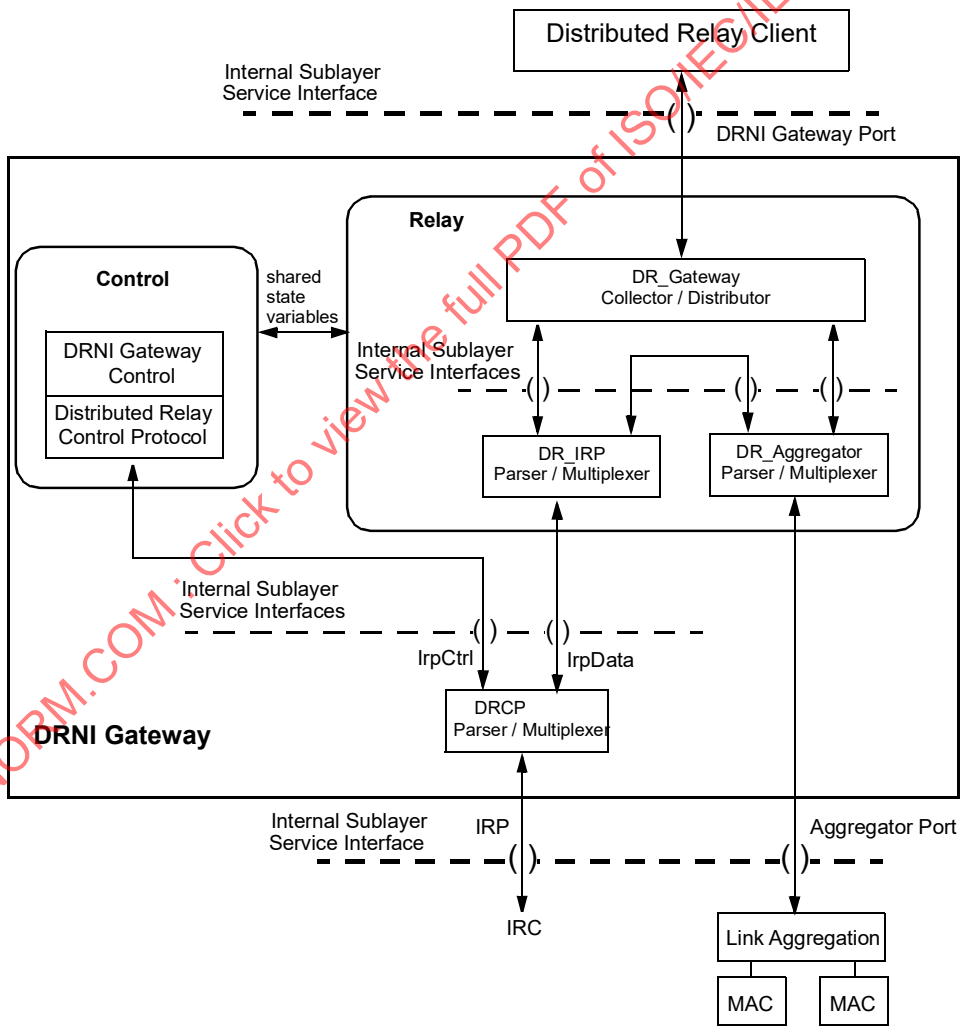



**Figure 9-7—DRNI Gateway block diagram**

The Distributed Relay passes every frame received from an Aggregator Port (Up frame) to a single DRNI Gateway Port, or discards it, and every frame received from a DRNI Gateway Port (Down frame) to an Aggregator Port, or discards it. The DRNI Gateways providing the Distributed Relay functionality sometimes have to send a frame across the IRC to reach the appropriate DRNI Gateway Port or Aggregator Port. The DRNI Gateway Relay forwards frames among the DRNI Gateway Port, Aggregator Port, or Intra-Relay Port by first mapping each frame to a Gateway Conversation ID and a Port Conversation ID. It then uses a set of Boolean masks, each indexed by one of these Conversation IDs, to determine whether the frame is permitted to pass through each of the Ports.

The DRCP Parser/Multiplexer on the IRP is responsible for forwarding Distributed Relay Control Protocol (DRCP) frames received at the IRP to the Control block and all other frames (collectively referred to as "data" frames) to the Relay block. The entities in the Relay block forward data frames among the DRNI Gateway, Aggregator, and Intra-Relay Ports. The entities in the Control block generate the Boolean masks used in the Relay, based upon local configuration and state as well as state obtained from the paired DRNI System via DRCP.

### 9.5.1 DRCP Parser/Multiplexer

The DRCP Parser/Multiplexer on the IRP is an instance of the Protocol Parser/Multiplexer described in 6.1.3 with the following characteristics:

a) The DownPort of the DRCP Parser/Multiplexer is the IRP interface as shown in Figure 9-7.
b) The ControlPort of the DRCP Parser/Multiplexer is the IrpCtrl interface as shown in Figure 9-7.
c) The DataPort of the DRCP Parser/Multiplexer is the IrpData interface as shown in Figure 9-7.
d) The IsControlFrame function returns a value of TRUE if the DA of the service primitive is equal to the DRNI_DA value, the msdu_type is the DRNI EtherType (Table 9-2), and the subtype is equal to the DRCP Subtype (Table 9-3).

### 9.5.2 DRNI Gateway Relay

The DRNI Gateway shall implement the DR_Gateway Collector/Distributor (Figure 9-8), DR_Aggregator Parser/Multiplexer (Figure 9-9), and DR_IRP Parser/Multiplexer (Figure 9-10), with their associated parameters (9.5.2.2 through 9.5.2.4). These state machines, in cooperation with the corresponding state machines in another DRNI Gateway with compatible configuration and connected via the IRC, implement the Distributed Relay operation behavior described in 9.2.

#### 9.5.2.1 Service interfaces

Since a DRNI Gateway uses various instances of the ISS, it is necessary to introduce a notation convention so that the reader can be clear about which interface is being referenced at any given time. A prefix is therefore assigned to each service primitive to indicate which of the interfaces is being invoked. The prefixes are as follows:

a) Aggregator:, for primitives issued on the interface between the DRNI Gateway and the Link Aggregation sublayer.
b) Gateway:, for primitives issued on the interface between the DRNI Gateway and the Distributed Relay Client.
c) IRP:, for primitives issued on the interface between the MAC (or Link Aggregation) entities supporting the IRC and the DRCP Parser/Multiplexer (9.5.1).
d) IrpCtrl:, for primitives issued on the interface between the DRCP Parser/Multiplexer and the DRNI Gateway control entity.
e) IrpData:, for primitives issued on the interface between the DRCP Parser/Multiplexer and the DRNI Gateway relay entity.

The Aggregator interface is a superset of the standard ISS. In addition to the standard ISS parameters and service primitives, the DRNI Gateway can access several of the Aggregator variables and can control the operational key of the Aggregator and Aggregation Ports, and changes to some of the Aggregator administrative and operational variables trigger actions in the DRNI Gateway.

#### 9.5.2.2 variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

BEGIN
: A Boolean variable that is set to TRUE when the System is initialized or reinitialized and is set to FALSE when (re)initialization has completed.
Data Type: Boolean

DWC
: An alias for the Home_Aggregator_State.Aggregator_CSCD_State.Oper_DWC value, which is derived from the Aggregator's Discard_Wrong_Conversation variable. When TRUE, an Up frame received at an Aggregator Port or IRP is discarded if a Down frame with the same Port Conversation ID would not be forwarded to the same Aggregator Port or IRP.
Data Type: Boolean

enableIrcData
: A signal from the Distributed Relay machine that the Home and Neighbor DRNI Gateways agree that the IRC can be used for passing data frames. The value of enableIrcData is TRUE when the IRC_Data, Sync, and DRNI values are all TRUE in both the Home_IRP_State and Neighbor_IRP_State variables; otherwise, enableIrcData is FALSE.
Data Type: Boolean

Home_Gateway_Mask
: Operational Boolean vector, indexed by Gateway Conversation ID, that indicates whether the indexed Gateway Conversation ID is allowed to pass through this DRNI Gateway's DRNI Gateway Port (TRUE = passes). Its values are computed by the updateDrMasks function (9.6.7).
Data Type: sequence of Boolean values, indexed by Gateway Conversation ID.
Alias of aDrniHomeGatewayConversationPasses (7.4.1.1.8).

Home_Aggregator_Mask
: Operational Boolean vector, indexed by Port Conversation ID, that indicates whether the indexed Port Conversation ID is allowed to be distributed through this DRNI Gateway's Aggregator Port (TRUE = passes). Its values are computed by the updateDrMasks function (9.6.7).
Data Type: sequence of Boolean values, indexed by Port Conversation ID.
Alias of aDrniHomeAggregatorConversationPasses (7.4.1.1.7).

Neighbor_Gateway_Mask
: Operational Boolean vector of which Gateway Conversation IDs are assigned to the DRNI Gateway Port reachable through the Intra-Relay Port. Its values are computed by the updateDrMasks function (9.6.7).
Data Type: sequence of Boolean flags indexed by Gateway Conversation ID.
For frames received on this IRP, TRUE means that the frame is a Down frame, destined for the Aggregator, and FALSE means the frame is an Up frame, destined for the DRNI Gateway Port.
Alias of aDrniNeighborGatewayConversationPasses (7.4.1.1.28).

Neighbor_Aggregator_Mask
: Operational Boolean vector of which Port Conversation IDs are assigned to the Aggregator Port reachable through the Intra-Relay Port. Its values are computed by the updateDrMasks function (9.6.7).
Data Type: sequence of Boolean flags indexed by Port Conversation ID.
Alias of aDrniNeighborAggregatorConversationPasses (7.4.1.1.27).

#### 9.5.2.3 Functions

extractGatewayConversationID
: This function determines the Gateway Conversation ID for the frame using the Home_Admin_Gateway_Algorithm, possibly in conjunction with the Home_Admin_Gateway_Conv_Service_Map, as described in 8.2.
Data Type: Integer.
Value: 0 to 4095.

extractPortConversationID
: This function determines the Port Conversation ID for the frame using the Aggregator's Actor_Port_Algorithm, possibly in conjunction with the Admin_Conv_Service_Map, as described in 8.2.
Data Type: Integer.
Value: 0 to 4095.

#### 9.5.2.4 Messages

Aggregator:M_UNITDATA.indication
Gateway:M_UNITDATA.indication
IrpData:M_UNITDATA.indication
Aggregator:M_UNITDATA.request
Gateway:M_UNITDATA.request
IrpData:M_UNITDATA.request
: The service primitives used to pass a received frame to a client with the specified parameters.

#### 9.5.2.5 DR_Gateway Collector/Distributor

The "Collector" function of the DR_Gateway Collector/Distributor forwards to the Gateway interface all indication service primitives passed to the DR_Gateway Collector/Distributor by the DR_Aggregator or DR_IRP state machines.

The "Distributor" function of the DR_Gateway Collector/Distributor is described by the DR_Gateway state machine shown in Figure 9-8. There is one DR_Gateway state machine per DRNI Gateway.

Invocation of a request service primitive at the Gateway interface triggers the DR_Gateway state machine. The state machine first maps the frame represented by the request to a Gateway Conversation ID and a Port Conversation ID.

If the Home_Gateway_Mask indicates that this is not the selected DRNI Gateway Port for this Gateway Conversation ID, then the frame is discarded. This prevents frames received from a DRNI Gateway Port from being forwarded across an IRC and passed back into the network through the DRNI Gateway Port of the paired DRNI System.

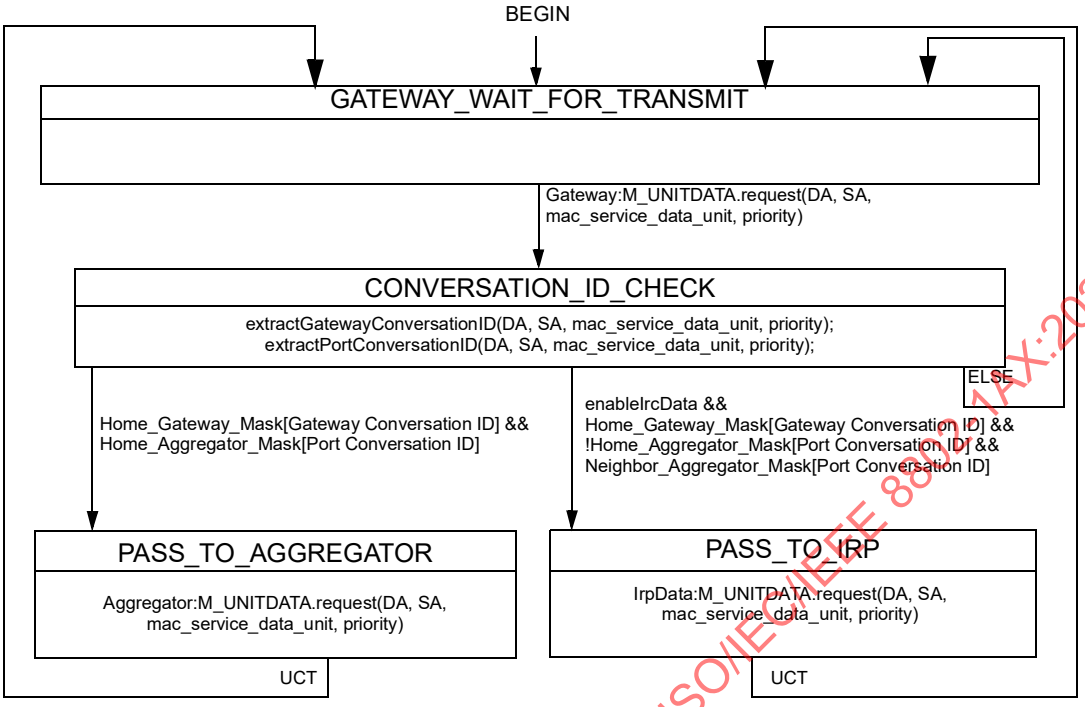



**Figure 9-8—DR_Gateway state machine**

If the frame is allowed to be received through this DRNI Gateway Port, then the following applies:

- If the Home_Aggregator_Mask indicates that the Aggregator Port in this DRNI Gateway is the selected port for this Port Conversation ID, then the frame is passed as a request service primitive to the DR_Aggregator Parser/Multiplexer.
- Otherwise, if the Neighbor_Aggregator_Mask indicates that the selected Aggregator Port for this is reachable through the IRC and if the transmission of data frames on the IRC has been enabled, then the frame is passed as a request service primitive to the DR_IRP Parser/Multiplexer.
- Otherwise, the frame is discarded.

### 9.5.2.6 DR_Aggregator Parser/Multiplexer

The “Multiplexer” function of the DR_Aggregator Parser/Multiplexer forwards to the Aggregator interface all request service primitives passed to the DR_Aggregator Parser/Multiplexer by the DR_Gateway or DR_IRP state machines.

The “Parser” function of the DR_Aggregator Parser/Multiplexer is described by the DR_Aggregator state machine shown in Figure 9-9. There is one DR_Aggregator state machine per DRNI Gateway.

Invocation of an indication service primitive at the Aggregator interface triggers the DR_Aggregator state machine. The state machine first maps the frame represented by the indication to a Gateway Conversation ID and a Port Conversation ID.

If the Home_Aggregator_Mask indicates that this is not the selected Aggregator Port for this Port Conversation ID and DWC is TRUE (indicating frames are received only on the Aggregator Port to which they would be distributed), then the frame is discarded.

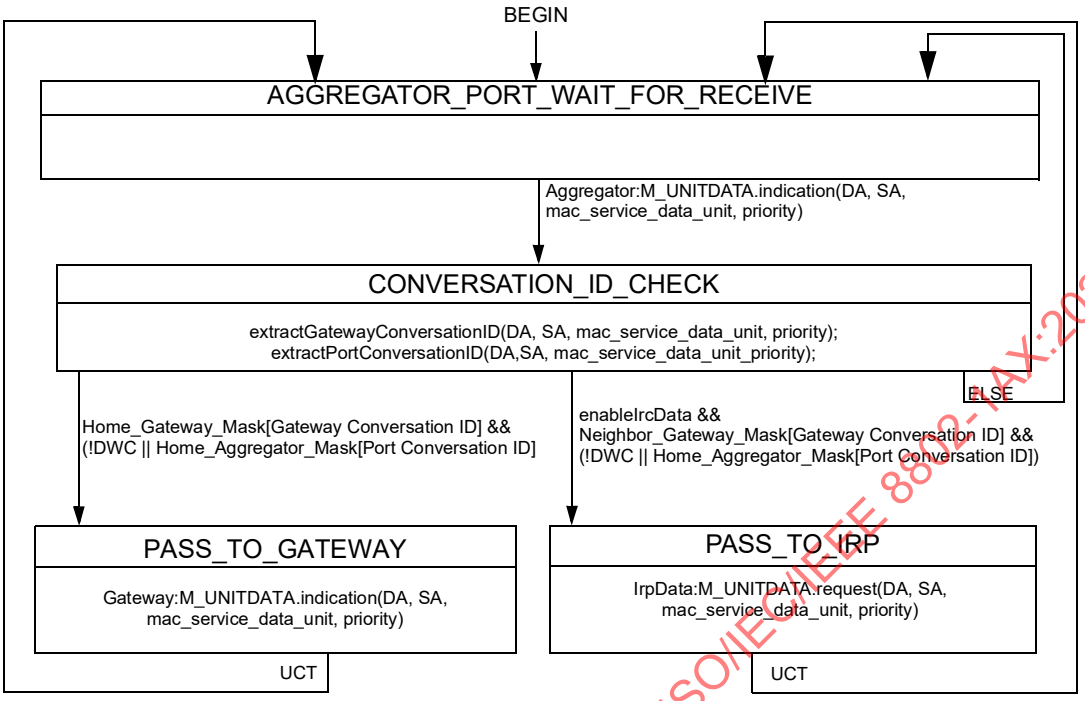



**Figure 9-9—DR_Aggregator state machine**

If the frame is allowed to be received through this Aggregator Port, then the following applies:

- If the Home_Gateway_Mask indicates that the DRNI Gateway Port in this DRNI Gateway is the selected port for this Gateway Conversation ID, then the frame is passed as an indication service primitive to the DR_Gateway Collector/Distributor.
- Otherwise, if the Neighbor_Gateway_Mask indicates that the selected DRNI Gateway Port for this is reachable through the IRC and if the transmission of data frames on the IRP has been enabled, then the frame is passed as a request service primitive to the DR_IRP Parser/Multiplexer.
- Otherwise, the frame is discarded.

### 9.5.2.7 DR_IRP Parser/Multiplexer

The “Multiplexer” function of the DR_IRP Parser/Multiplexer forwards to the IrpData interface all request service primitives passed to the DR_IRP Parser/Multiplexer by the DR_Gateway or DR_Aggregator state machines.

The “Parser” function of the DR_IRP Parser/Multiplexer is described by the DR_IRP state machine shown in Figure 9-10. There is one DR_IRP state machine in a DRNI Gateway.

Invocation of an indication service primitive at the IrpData interface triggers the DR_IRP state machine. The state machine first maps the frame represented by the indication to a Gateway Conversation ID and a Port Conversation ID.

If the Home_Gateway_Mask indicates that this is not the selected DRNI Gateway Port for this Gateway Conversation ID and if the Neighbor_Gateway_Mask indicates that the selected DRNI Gateway Port is reachable through this IRP, then the frame is a Down frame and is passed as a request service primitive to the DR_Aggregator Parser/Multiplexer.

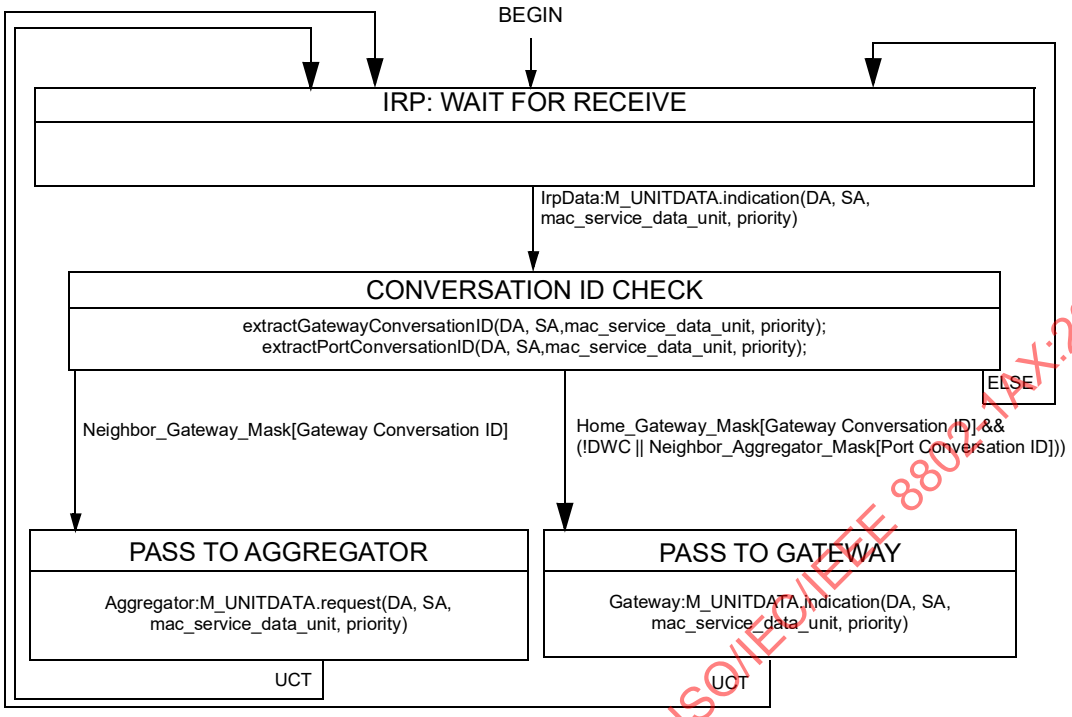



**Figure 9-10—DR_IRP state machine**

If the Home_Gateway_Mask indicates that this is the selected DRNI Gateway Port for this Gateway Conversation ID, then the frame is an Up frame. Furthermore, if the Home_Aggregator_Mask indicates that this is the selected Aggregator Port for this Port Conversation ID or DWC is FALSE, then the frame is passed as an indication service primitive to the DR_Gateway Collector/Distributor; otherwise, the frame is discarded.

### 9.5.3 DRNI Gateway Control

DRNI Gateway Control configures and controls the DRNI Gateway using static information local to the DRNI Gateway and the Aggregator and using dynamic information exchanged by means of the Distributed Relay Control Protocol, as described in 9.5.3.1 through 9.5.3.5.

#### 9.5.3.1 DRNI Gateway and DRNI Identification

A DRNI Gateway is uniquely identified by the combination of two System Identifiers, each with different uniqueness requirements:

Home System Identifier
: The identifier of this DRNI System, formed by the concatenation of the Actor_Admin_System_Priority and Actor_Admin_System of the Aggregator supporting this DRNI Gateway. The MAC address portion of this identifier can be shared with other functional elements within the same DRNI System, but is otherwise unique among any set of systems with potential links to each other. This identifier and the Actor_Admin_Aggregator_Key are used for the Actor_Oper_System_Priority, Actor_Oper_System, and Actor_Oper_Aggregator_Key values in LACPDUs transmitted by the Aggregation Ports assigned to this DRNI Gateway when the DRNI Gateway is not paired with a neighbor.

DRNI Aggregator Identifier
: The identifier of the DRNI created when this DRNI Gateway is paired with a neighbor via the IRC and formed by the concatenation of the DRNI_Aggregator_System_Priority and DRNI_Aggregator_System variables associated with this DRNI Gateway. These values are configured to be the same in the DRNI Gateways of both DRNI Systems that constitute the DRNI, and they can be zero or can be the same as the Home System Identifier in one of the DRNI Systems. The MAC address portion of this identifier is otherwise unique among the set of Aggregators and/or DRNI Gateways (intended to support different Distributed Relays) within each DRNI System and among any set of systems with potential links to each other. This identifier and the DRNI_Aggregator_Key determine the Actor_Oper_System_Priority, Actor_Oper_System, and Actor_Oper_Aggregator_Key values in LACPDUs transmitted by the Aggregation Ports assigned to this DRNI Gateway when the DRNI Gateway is paired with a neighbor.

The DRNI Gateway connected to this DRNI Gateway by an IRC is identified by the following:

Neighbor System Identifier
: The identifier of the Neighbor DRNI System, formed by the concatenation of the Neighbor_System_Priority and Neighbor_System values. These values are obtained from DRCPDUs received at the IRP.

#### 9.5.3.2 Forming the DRNI

A DRNI Gateway operating independently (i.e., not having formed a DRNI with another DRNI Gateway) sets the operational values of the actor identifier and key for the Aggregator and all Aggregation Links to the administrative values for the Aggregator.

When the IRC is operational (i.e., the IRP ISS MAC_Operational value is TRUE), the DRNI Gateway transmits DRCPDUs and awaits the receipt of a DRCPDU from a DRNI Gateway with a compatible configuration. Upon receipt of a DRCPDU from a DRNI Gateway with a different Home System Identifier but the same DRNI Aggregator Identifier and the same DRNI Aggregator Key, the two DRNI Gateways form a DRNI. Both DRNI Gateways change the operational actor identifier values (for the Aggregator and all Aggregation Links configured to be part of the DRNI) to the DRNI_Aggregator_System_Priority and DRNI_Aggregator_System values or, if the DRNI_Aggregator_System is zero, to the numerically lowest of the Home System Identifier and the Neighbor System Identifier. Likewise both DRNI Systems change the operational actor key value (for the Aggregator and all Aggregation Links configured to be part of the DRNI) to the DRNI_Aggregator_Key value or, if the DRNI_Aggregator_System value is zero, to the Actor_Admin_Aggregator_Key value of the DRNI System with the numerically lowest system identifier.

#### 9.5.3.3 Partner Selection and Forming a LAG

When a DRNI has been formed, all active links in the DRNI LAG connect to the same LACP Partner, as indicated by the Partner System Identifier and Key values. If the DRNI System with the lowest Home System Identifier has any active links, the other DRNI System will select the DRNI Aggregator only for Aggregation Ports connected to the same LACP Partner and will set Selected to UNSELECTED for any Aggregation Ports connected to a different LACP Partner.

#### 9.5.3.4 Port Algorithm and Aggregator Port Selection

Each DRNI Gateway in a DRNI has an Aggregator Port supported by an Aggregator in a Link Aggregation sublayer. The DRNI Gateway uses the Aggregator's CSCD parameters, particularly the Actor_Port_Algorithm, Admin_Conv_Service_Map, and Admin_Conv_Link_Map, together with the list of Active_LAG_Links from both DRNI Gateways, to determine which Aggregation Link is used to forward

frames received at the DRNI Gateway Port, as described in 6.6.1. The selected Link_Number determines the selected Aggregator Port.

The DRNI Gateway transmits the Aggregator's CSCD parameters, LACP Partner identification, and the list of Active_LAG_Links in the Aggregator State TLV in DRCPDUs. It receives the Neighbor_Aggregator_State and an acknowledgment of this DRNI Gateway's Aggregator_Sequence_Number in DRCPDUs sent by the Neighbor. The Home_Aggregator_State and Neighbor_Aggregator_State are used to determine the Selected_Aggregator_Vector that contains the Aggregator Port selected by the DRNI Gateway for each Port Conversation ID. This is used to determine the Home_Aggregator_Mask and Neighbor_Aggregator_Mask used by the DR_Gateway Collector/Distributor to forward frames from the DRNI Gateway Port to the Aggregator Port or IRP. When the Neighbor DRNI Gateway's and the LACP Partner's CSCD parameters match this Aggregator's parameters and when the Aggregator's Discard_Wrong_Conversation variable is TRUE, then the Home_Aggregator_Mask and Neighbor_Aggregator_Mask are also used to validate the forwarding of frames from an Aggregator Port or IRP to the DRNI Gateway Port.

#### 9.5.3.5 Gateway Algorithm and DRNI Gateway Port selection

Each DRNI Gateway has a DRNI Gateway Port that connects to the Distributed Relay Client in the DRNI System. For any given frame traversing a DRNI LAG, one and only one of the two DRNI Gateway Ports is selected to forward that frame. This DRNI Gateway Port is determined by first using the Gateway Algorithm to map the frame to a Gateway Conversation ID and then using the results of the DRNI Gateway Port selection to map the Gateway Conversation ID to one of the DRNI Gateway Ports.

The Gateway Algorithm, possibly in conjunction with an associated Gateway Service ID map, is used as specified in 8.2 to associate frames with Gateway Conversation IDs. Both DRNI Gateways need to use the same algorithm and map to assure that both DRNI Gateways associate a given frame with the same Gateway Conversation ID. This is necessary to guarantee that the DRNI Gateway in each DRNI System would select the same DRNI Gateway Port for a given frame, unless one DRNI Gateway Port is selected for all Gateway Conversation IDs (and therefore all frames). Therefore, if the Gateway Algorithm and Gateway Service ID map are not the same in both DRNI Gateways, only the DRNI Gateway Port in the DRNI Gateway with the lowest Home System Identifier is eligible to be selected for any Gateway Conversation ID.

NOTE—If the DRNI System is a VLAN Bridge performing learning, the Gateway Algorithm mapping a frame to a Gateway Conversation ID can be based on its VLAN ID to avoid oscillations in the learning process within the attached network. For implementations of DRNI in these cases, there can be a one-to-one map of VLAN ID to Gateway Conversation ID.

DRNI Gateway Port selection is a mechanism for the DRNI Gateways to select one and only one of the DRNI Gateway Ports to forward frames mapping to each Gateway Conversation ID. A variety of DRNI Gateway Port selection policies are supported, including the following:

— **Administrative Control**—The Home_Admin_Gateway_Enable_Mask in each DRNI Gateway provides separation of frames traversing the DRNI Gateways into discarded, unprotected, and protected conversations. When neither DRNI Gateway has a Gateway Conversation ID enabled, frames mapping to that Gateway Conversation ID are not forwarded through either DRNI Gateway Port (discarded conversations). When only one DRNI Gateway has a Gateway Conversation ID enabled, then only the DRNI Gateway Port in that DRNI Gateway is eligible to be selected (unprotected conversations). When both DRNI Gateways have a Gateway Conversation ID enabled, either DRNI Gateway Port is eligible for selection (protected conversations). For protected conversations, the final DRNI Gateway Port selection is determined by a combination of Distributed Relay Client Control (if that policy is enabled), Link Aggregation Control (if that policy is enabled), and the Home_Admin_Gateway_Preference_Mask vector in each DRNI Gateway.

— **Distributed Relay Client Control**—When administratively enabled by setting Home_Admin_Client_Gateway_Control to TRUE, the Distributed Relay Client can determine whether frames can be forwarded through the DRNI Gateway Port, by controlling the value of the Gateway_Available_Mask for any Gateway Conversation ID enabled by the Home_Admin_Gateway_Enable_Mask. The specifics of how the Distributed Relay Client makes this determination are dependent upon the specific functionality of the Distributed Relay Client and beyond the scope of this standard. Possible examples include the following:

  **Protection Switching**: The conversation is protected within the network by a Protection Switching scheme where the Working Path and Protection Path are routed to the Distributed Relay operation through different Gateway Ports.
  **VLAN membership**: When the Distributed Relay Client is a Bridge and the conversations correlate to VLAN Identifiers, then the VLAN Member Set information in the Filtering Database can be used to determine the eligibility of the DRNI Gateway Port for each Gateway Conversation ID.

— **Link Aggregation Control**—When administratively enabled by setting Home_Admin_CSCD_Gateway_Control to TRUE, the Aggregator's CSCD configuration that determines Aggregator and Aggregation Port selection also determines the DRNI Gateway Port selection so that the selected DRNI Gateway and Aggregation Ports are in the same DRNI System. This requires that the Gateway Algorithm and Port Algorithm be the same. The result is to minimize data traffic on the IRC at the expense of allowing Aggregation Link failure and recovery events to affect the data forwarding topology in the network.

The DRNI Gateway transmits the gateway configuration and state information in the Gateway State TLV and Gateway Preference TLV in DRCPDUs. It receives the Neighbor_Gateway_State and Neighbor_Gateway_Preference, together with an acknowledgment of this DRNI Gateway's Gateway_Sequence_Number and Gateway_Preference_Sequence_Number, in DRCPDUs sent by the Neighbor. The Home and Neighbor gateway information is used to determine this DRNI Gateway's Gateway Port selection for each Gateway Conversation ID (Selected_Gateway_Vector).

The DRNI Gateway maintains a Boolean synchronization variable for each Gateway Conversation ID. This is set to FALSE when there has been any change to the home state that could potentially affect the gateway selection for that Gateway Conversation ID and is set to TRUE when the DRNI Gateway is fully synchronized with the Neighbor (all reflected sequence numbers match the home sequence numbers). If there is a change in the Gateway Port selection for any Gateway Conversation ID when the synchronization variable is FALSE, both the Home_Gateway_Mask and the Neighbor_Gateway_Mask will be set to FALSE for that Gateway Conversation ID. Otherwise, the Home_Gateway_Mask is set to TRUE when this DRNI Gateway has been selected, and the Neighbor_Gateway_Mask is set to TRUE when the Neighbor DRNI Gateway has been selected. These masks are used by the DRNI Gateway Relay state machines to determine whether frames can be received from or forwarded to the DRNI Gateway Port and to distinguish Up from Down frames received at the IRP.

## 9.6 Distributed Relay Control Protocol

The purpose of the Distributed Relay Control Protocol (DRCP) is to

a) Establish communication between DRNI Gateways across an IRC.
b) Verify the consistent configuration of DRNI Gateways.
c) Distribute the current states of the DRNI Gateways and their Aggregators among each other.
d) Compute the resultant path of any frame required to pass through the IRC, and exchange the information with the Neighbor DRNI Gateway as required to ensure against forwarding loops and duplicate frame delivery.
e) Maintain the DRNI LAG across the paired DRNI Systems.

The result of the operation of DRCP is to

f) Determine whether the DRNI Gateway will operate independently or as part of a DRNI (9.5.3.1 and 9.5.3.2).
g) Maintain the operational key value of the Aggregator and maintain the DRNI LAG with the selected Partner System or DRNI (9.5.3.2 and 9.5.3.3).
h) Maintain the Boolean masks that control the forwarding of frames by the DRNI Gateway (9.5.2.2, 9.5.3.4, and 9.5.3.5).

### 9.6.1 DRCPDU transmission, addressing, and protocol identification

Distributed Relay Control Protocol Data Units (DRCPDUs) are transmitted and received using the service provided by an LLC entity that uses, in turn, a single instance of the MAC Service provided at an MAC Service Access Point (MSAP) associated with an IRP. Each DRCPDU is transmitted as a single MAC service request, and received as a single MAC service indication, with the following parameters:

a) Destination address (9.6.1.1)
b) Source address (9.6.1.2)
c) MAC Service Data Unit (MSDU)
d) Priority (9.6.1.3)

The MSDU of each request and indication comprises a number of octets that provide EtherType protocol identification (9.6.1.5) followed by the DRCPDU proper (9.6.2).

NOTE 1—For the purposes of this standard, the term "LLC entity" includes entities that support protocol discrimination using the EtherType field as specified in IEEE Std 802.

NOTE 2—The complete format of an DRCP frame depends not only on the DRCPDU format, as specified in this clause, but also on the media access method-dependent procedures used to support the MAC Service.

#### 9.6.1.1 Destination MAC Address

The destination address (DA) for each MAC service request used to transmit a DRCPDU is configured in the DRNI_DA variable. The group addresses listed in Table 9-1 can be used to determine the scope of propagation of DRCPDUs within a Bridged LAN (see Tables 8-1, 8-2, and 8-3 of IEEE Std 802.1Q-2018),

**Table 9-1—Distributed Relay Control Protocol destination addresses**

| Assignment | Value |
|---|---|
| Nearest Customer Bridge group address | 01-80-C2-00-00-00 |
| Nearest Bridge group address | 01-80-C2-00-00-0E |
| Nearest non-TPMR Bridge group address | 01-80-C2-00-00-03 |

The default value is the Nearest non-TPMR Bridge group address.

NOTE—The destination address used in DRCPDUs is not restricted to being one of the addresses in Table 9-1; therefore, the use of other suitable address scopes is permitted without further revision of this standard.

#### 9.6.1.2 Source MAC Address

The source address (SA) for each MAC service request used to transmit a DRCPDU shall be an individual address associated with the IRP MSAP at which the request is made.

#### 9.6.1.3 Priority

The priority associated with each MAC Service request is the default associated with the IRP MSAP.

#### 9.6.1.4 Protocol Identification

All DRNI protocols use the Distributed Resilient Network Interconnect EtherType value as the primary means of protocol identification; its value is shown in Table 9-2.

**Table 9-2—DRNI EtherType Assignment**

| Assignment | Value |
|---|---|
| DRNI EtherType | 89–52 |

The first octet of the MAC Client data following the EtherType field is a protocol subtype identifier that distinguishes between different DRNI protocols. Table 9-3 identifies the semantics of this subtype.

**Table 9-3—DRNI Protocol subtypes**

| Protocol Subtype value | Protocol name |
|---|---|
| 0 | Reserved for future use |
| 1 | Distributed Relay Control Protocol (DRCP) (9.6) |
| 2 | Deprecated |
| 3–255 | Reserved for future use |

#### 9.6.1.5 Encapsulation of DRCPDUs in frames

A DRCPDU is encoded in the mac_service_data_unit parameter of an M_UNITDATA.request or M_UNITDATA.indication. The first octets of the mac_service_data_unit are the DRNI EtherType (Table 9-2), followed by the DRCPDU as specified in 9.6.2.3.

### 9.6.2 DRCPDU structure and encoding

#### 9.6.2.1 Transmission and representation of octets

All DRCPDUs comprise an integral number of octets. The octets in a DRCPDU are numbered starting from 1 and increasing in the order they are put into the DRCPDU. The bits in each octet are numbered from 1 to 8, where bit 1 is the least significant bit.

When consecutive bits within an octet are used to represent a binary number, the highest bit number has the most significant value. When consecutive octets are used to represent a binary number, the most significant

octet has the lowest octet number and is transmitted first, followed by successively less significant octets (with successively higher octet numbers).

When the encoding of (an element of) a DRCPDU is depicted in a diagram,

a) Elements of an DRCPDU are transmitted from top to bottom.
b) When an element contains multiple octets, the most significant octet is transmitted first.
c) When multiple octets are depicted side by side, the octets are transmitted from left to right.
d) Within an octet, bits are shown with bit 8 (the most significant bit) to the left and bit 1 (the least significant bit) to the right.

### 9.6.2.2 DRCP TLV structure

The basic TLV format, applicable to all DRCP TLVs, is shown in Figure 9-11.

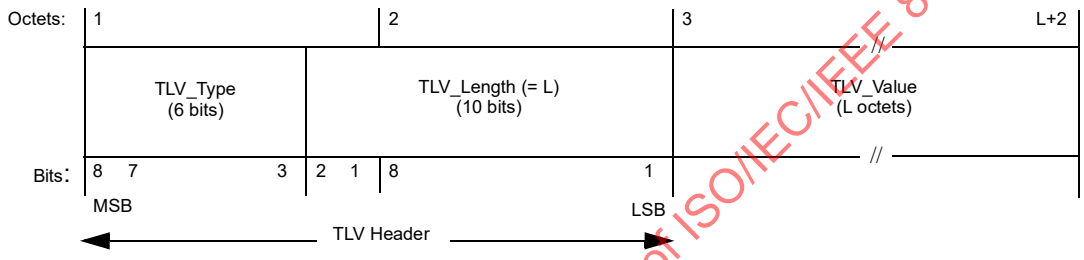



**Figure 9-11—Basic TLV format**

The TLV_Type field occupies the six most significant bits of the first octet of the TLV format. The two least significant bits in the first octet of the TLV format are the two most significant bits of the TLV_Length field.

NOTE—The DRCP TLV_Length fields provide the length of the TLV_Value fields in contrast to the LACP TLV_Length fields that provide the length of the total TLV (including the 2 octets for the TLV_Type and TLV_Length field). This enables the use of TLVs carrying a TLV_Value field of a length up to 1023 octets.

Table 9-4 provides a list of the TLVs that are applicable for DRCP.

**Table 9-4—Type field values of DRCP TLVs**

| Type field | TLV | Support |
|---|---|---|
| 0 | Terminator | Mandatory (9.6.2.3) |
| 1 | Home System Identification | Mandatory (9.6.2.3) |
| 2 | Neighbor System Identification | Mandatory (9.6.2.3) |
| 3 | DRNI State | Mandatory (9.6.2.3) |
| 4 | Aggregator State | Mandatory (9.6.2.4) |
| 5 | Gateway State | Mandatory (9.6.2.5) |
| 6 | Gateway Preference | Mandatory (9.6.2.6) |
| 7 | Organization-Specific | Optional (9.6.2.7) |
| 8–63 | Reserved for IEEE 802.1 | |

### 9.6.2.3 DRCPDU structure

The minimum DRCPDU structure, as generated by the DRCP Transmit machine (9.6.13), is shown in Figure 9-12 and further described in the field definitions after the figure. The structure and field definitions of other TLVs that can be included in the DRCPDU are described in 9.6.2.4 through 9.6.2.7. Requirements regarding the validation and processing of received DRCPDUs are specified in the DRCP Receive machine (9.6.10).

| | Length (in octets) |
|---|---|
| **Subtype = DRCP** | **1** |
| **Version Number = 2** | **1** |
| **TLV_Type = Home_System_Identification** | **2** |
| Home_System_Identification_Length = 18 | |
| Home_System_Priority | 2 |
| Home_System | 6 |
| DRNI_Aggregator_System_Priority | 2 |
| DRNI_Aggregator_System | 6 |
| DRNI_Aggregator_Key | 2 |
| **TLV_Type = Neighbor_System_Identification** | **2** |
| Neighbor_System_Identification_Length = 8 | |
| Neighbor_System_Priority | 2 |
| Neighbor_System | 6 |
| **TLV_Type = DRNI_State** | **2** |
| DRNI_State_Length = 26 | |
| Home_Aggregator_Sequence_Number | 4 |
| Neighbor_Aggregator_Sequence_Number | 4 |
| Home_Gateway_Sequence_Number | 4 |
| Neighbor_Gateway_Sequence_Number | 4 |
| Home_Gateway_Preference_Sequence_Number | 4 |
| Neighbor_Gateway_Preference_Sequence_Number | 4 |
| Home_IRP_State | 1 |
| Neighbor_IRP_State | 1 |
| **Other TLVs** | **n** |
| **TLV_Type = Terminator** | **2** |
| **Terminator_Length = 0** | |

**Figure 9-12—DRCPDU structure**

a) *Subtype.* The Subtype field identifies the specific Protocol being encapsulated. DRCPDUs carry the Subtype value 0x01.

b) *Version Number.* This identifies the DRCP version; implementations conformant to this standard carry the value 0x02.

c) ***TLV_Type = Home System Identification***. Value is 0x01. This TLV includes the information necessary to identify the DRNI and DRNI System and to validate the topology of the DRNI.

d) *Home_System_Identification_Length.* Value is 18.

e) *Home_System_Priority*. The Actor_Admin_System_Priority value of the Aggregator supporting the DRNI Gateway. The value indicates the priority component of the System Identifier of this DRNI System.

f) *Home_System*. The Actor_Admin_System value of the Aggregator supporting the DRNI Gateway. The value indicates the MAC Address component of the System Identifier of this DRNI System.

g) *DRNI_Aggregator_System_Priority.* The DRNI_Aggregator_System_Priority value assigned to this DRNI Gateway.

h) *DRNI_Aggregator_System.* The DRNI_Aggregator_System value assigned to this DRNI Gateway.

i) *DRNI_Aggregator_Key*. The value to be used as the actor key in LACPDUs on this DRNI. When Home_IRP_State.Sync is TRUE and the Neighbor System Identifier is numerically lower than the Home System Identifier, this value is equal to the Neighbor_DRNI_Key variable; otherwise, it is equal to the DRNI_Aggregator_Key variable.

j) ***TLV_Type = Neighbor System Identification***. Value is 0x02. This TLV echoes the *Home System Identification* information received in a DRCPDU from the Neighbor.

k) *Neighbor System Identification Length*. Value is 8.

l) *Neighbor_System_Priority*. The Neighbor_System_Priority value, indicating the priority component of the System Identifier of the immediate Neighbor on this IRP.

m) *Neighbor_System*. The Neighbor_System value, indicating the MAC Address component of the System Identifier of the immediate Neighbor on this IRP.

n) ***TLV_Type = DRNI_State***. Value is 0x03. This TLV contains the Gateway and Aggregator Sequence Numbers, and the IRP_State, for the Home and Neighbor DRNI Gateways.

o) *DRNI State Length*. Value is 26.

p) *Home Aggregator Sequence Number.* The Aggregator Sequence Number from the Home_Aggregator_State.

q) *Neighbor Aggregator Sequence Number.* The Aggregator Sequence Number from the Neighbor_Aggregator_State.

r) *Home Gateway Sequence Number*. The Gateway Sequence Number from the Home_Gateway_State.

s) *Neighbor Gateway Sequence Number.* The Gateway Sequence Number from the Neighbor_Gateway_State.

t) *Home Gateway Preference Sequence Number*. The Gateway Preference Sequence Number from the Home_Gateway_Preference.

u) *Neighbor Gateway Preference Sequence Number*. The Gateway Preference Sequence Number from the Neighbor_Gateway_Preference.

v) *Home IRP State.* This DRNI Gateway's topology-related variables for the IRP, encoded within a single octet, as illustrated in Figure 9-13 and further defined after the figure. All Boolean values are encoded as 1 for TRUE and 0 for FALSE.

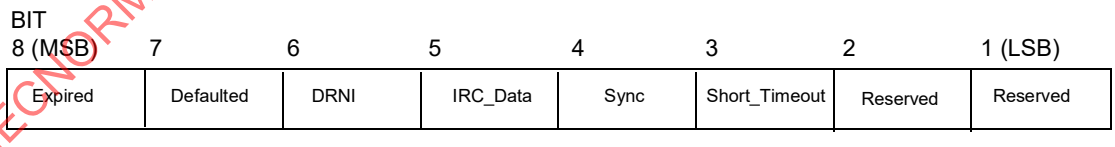



**Figure 9-13—Bit encoding of the Home_IRP_State and Neighbor_IRP_State fields**

1) Bit 1 is reserved for future use. It is set to 0 on transmit and ignored on receipt.
2) Bit 2 is reserved for future use. It is set to 0 on transmit and ignored on receipt.
3) *Short_Timeout* is encoded in bit 3. This flag indicates the Timeout control value in use by the DRCP Receive machine on this IRP. Short Timeout is encoded as a 1; Long Timeout is encoded as a 0.

4) *Sync* is encoded in bit 4. When this flag is TRUE, the DRCP Receive machine has determined the Neighbor DRNI System has a compatible configuration for forming a DRNI.
5) *IRC_Data* is encoded in bit 5. When this flag is TRUE, the transfer of Up and Down frames is permitted on the IRC.
6) *DRNI* is encoded in bit 6. This flag is TRUE when this DRNI System is paired with another DRNI System (i.e., when DR_Solo is FALSE) and FALSE otherwise.
7) *Defaulted* is encoded in bit 7. When this flag is TRUE, the DRCP Receive machine is using default operational Neighbor information. When FALSE, the operational Neighbor information in use has been received in a DRCPDU.
8) *Expired* is encoded in bit 8. When this flag is TRUE, the DRCP Receive machine is in the EXPIRED state.

w) *Neighbor_IRP_State*. The contents of this field echo the *Home_IRP_State* received in a DRCPDU from the Neighbor, encoded within a single octet as illustrated in Figure 9-13.

x) **Other TLVs**. This field contains additional TLVs (e.g., 9.6.2.4 to 9.6.2.7) that are not required in every DRCPDU.

y) ***TLV_Type* = *Terminator***. This field indicates the nature of the information carried in this TLV-tuple. Terminator (end of message) information is identified by the integer value 0.

z) *Terminator_Length*. The length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).

NOTE—A Version 2 implementation is not intended to interoperate with a Version 1 implementation. The DRCP Receive machine specified in this standard discards received Version 1 DRCPDUs. A Version 1 implementation discards a received Version 2 DRCPDU because it fails the test specified in the PORTAL_CHECK state in Figure 9-23 of IEEE Std 802.1AX-2014.

### 9.6.2.4 Aggregator State TLV

The Aggregator State TLV is included in the DRCPDU when the value of the Aggregator_Sequence_Number in the Home_Aggregator_State differs from the value in the Reflected_Aggregator_Sequence_Number. The TLV is transmitted with the values in the Home_Aggregator_State variable (9.6.5). The Aggregator State TLV structure is shown in Figure 9-14 and further described in the field definitions after the figure.

| | |
|---|---|
| TLV_Type = Aggregator State | 2 |
| Aggregator_State_Length = 52 + (2 * n) | |
| Aggregator_Sequence_Number | 4 |
| Aggregator_Port_Algorithm | 4 |
| Aggregator_Conv_Service_Digest | 16 |
| Aggregator_Conv_Link_Digest | 16 |
| Partner_System_Priority | 2 |
| Partner_System | 6 |
| Partner_Oper_Aggregator_Key | 2 |
| Aggregator_CSCD_State | 1 |
| Reserved | 1 |
| Active_LAG_Links | 2 * n |

**Figure 9-14—Aggregator State TLV**

a) ***TLV_Type = Aggregator State***; Value as shown in Table 9-4.

b) *Aggregator State Length*; Value is 52 octets plus two octets for each Link_Number in the *Active_LAG_Links* field.

c) *Aggregator Sequence Number*. The Aggregator Sequence Number representing the current set of values for the remaining variables in this TLV.

d) *Aggregator Port Algorithm*. The Actor_Port_Algorithm used by the Aggregator and the DRNI Gateway to assign frames to Port Conversation IDs.

e) *Aggregator_Conv_Service_Digest*. A MD5 digest of the Admin_Conv_Service_Map used by the Aggregator and the DRNI Gateway to assign frames to Port Conversation IDs. The value can be transmitted as all zeros when the Actor_Port_Algorithm has the value "Unspecified" (Table 8-1) or when the Actor_Port_Algorithm does not use the Admin_Conv_Service_Map.

f) *Aggregator_Conv_Link_Digest*. A MD5 digest of the Admin_Conv_Link_Map used by the Aggregator and the DRNI Gateway to map Port Conversation IDs to Aggregation Links. The value can be transmitted as all zeros when the Actor_Port_Algorithm has the value "Unspecified" (Table 8-1).

g) *Partner_System_Priority*. The Aggregator's Partner_Oper_System_Priority.

h) *Partner_System*. The Aggregator's Partner_Oper_System.

i) *Partner_Oper_Aggregator_Key*. The Aggregator's Partner_Oper_Aggregator_Key.

j) *Aggregator_CSCD_State*. Flags summarizing the Conversation-Sensitive Collection and Distribution (CSCD) state of the Aggregator, encoded within a single octet, as illustrated in Figure 9-15 and further defined after the figure. All Boolean values are encoded as 1 for TRUE and 0 for FALSE.

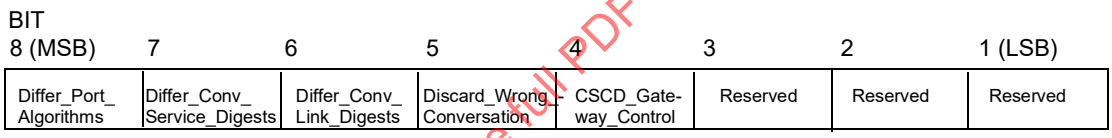



**Figure 9-15—Bit encoding of the Aggregator_CSCD_State fields**

1) Bits 1, 2, and 3 are reserved for future use. They are set to 0 on transmit and ignored on receipt
2) CSCD_Gateway_Control is encoded in bit 4. When this flag is TRUE, the DRNI Gateway is configured to minimize forwarding data frames on the IRC by selecting the DRNI Gateway and Aggregator Ports for forwarding any given Conversation ID to be in the same DRNI System.
3) *Discard_Wrong_Conversation* is encoded in bit 5. The Aggregator's Discard_Wrong_Conversation value.
4) *Differ_Conv_Link_Digests* is encoded in bit 6. The Aggregator's differConvLinkDigests flag is TRUE when the Aggregator's Actor_Conv_Link_Digest matches the Aggregator's Partner_Conv_Link_Digest.
5) *Differ_Conv_Service_Digests* is encoded in bit 7. The Aggregator's differConvServiceDigests flag is TRUE when the Aggregator's Actor_Conv_Service_Digest matches the Aggregator's Partner_Conv_Service_Digest.
6) *Differ_Port_Algorithms* is encoded in bit 8. The Aggregator's differPortAlgorithms flag is TRUE when the Aggregator's Actor_Port_Algorithm matches the Aggregator's Partner_Port_Algorithm.

k) *Reserved*. This octet is reserved for future use. It is set to 0 on transmit and ignored on receipt.

l) *Active_LAG_Links*. A list, perhaps empty, of the Link_Numbers of any Aggregation Links currently active on the Aggregator.

### 9.6.2.5 Gateway State TLV

The Gateway State TLV is included in the DRCPDU when the value of the Gateway_Sequence_Number in the Home_Gateway_State differs from the value in the Reflected_Gateway_Sequence_Number. The TLV is transmitted with the values in the Home_Gateway_State variable (9.6.5). The Gateway State TLV structure is shown in Figure 9-16 and further described in the field definitions after the figure.

| | |
|---|---|
| TLV_Type = Gateway State | 2 |
| Gateway_State_Length = 552 | |
| Gateway_Sequence_Number | 4 |
| Gateway_Algorithm | 4 |
| Gateway_Conv_Service_Digest | 16 |
| Gateway_Available_Mask | 512 |

**Figure 9-16—Gateway State TLV**

a) ***TLV_Type = Gateway State***; Value as shown in Table 9-4.
b) *Gateway_State_Length*; Value is 552.
c) *Gateway_Sequence_Number*. The Gateway Sequence Number representing the current set of values for the remaining variables in this TLV.
d) *Gateway_Algorithm*. The gateway algorithm administratively assigned for use by the DRNI Gateway to assign frames to Gateway Conversation IDs.
e) *Gateway_Conv_Service_Digest*. A MD5 digest of an array, indexed by Gateway Conversation ID, where each entry is a set of Service IDs (8.2) that map to that Gateway Conversation ID. The value can be transmitted as all zeros when the gateway algorithm has the value “Unspecified” (Table 8-1) or when the gateway algorithm does not use the Home_Admin_Gateway_Conv_Service_Map.
f) *Gateway_Available_Mask.* A Boolean vector, indexed by Gateway Conversation ID, that indicates whether the DRNI Gateway Port is currently able to forward frames for each Gateway Conversation ID.

### 9.6.2.6 Gateway Preference TLV

The Gateway Preference TLV is included in the DRCPDU when the value of the Gateway_Preference_Sequence_Number in the Home_Gateway_Preference differs from the value in the Reflected_Gateway_Preference_Sequence_Number. The TLV is transmitted with the values in the Home_Gateway_Preference variable (9.6.5). The Gateway Preference TLV structure is shown in Figure 9-17 and further described in the field definitions after the figure.

| | |
|---|---|
| TLV_Type = Gateway Preference | 2 |
| Gateway_Preference_Length = 514 | |
| Gateway_Preference_Sequence_Number | 2 |
| Gateway_Preference_Mask | 512 |

**Figure 9-17—Gateway Preference TLV**

a) *TLV_Type = Gateway Preference.* Value is shown in Table 9-4.
b) *Gateway_Preference_Length.* Value is 514.
c) *Gateway_Preference_Sequence_Number*. The Gateway Preference Sequence Number representing the current set of values for the remaining variable in this TLV.
d) *Gateway_Preference_Mask*. A Boolean vector, indexed by Gateway Conversation ID, that indicates whether the DRNI Gateway Port is the preferred Gateway to forward frames for each Gateway Conversation ID.

### 9.6.2.7 Organization-Specific TLV

These optional Organization-Specific TLVs are provided to allow different organizations, such as IEEE, ITU-T, and IETF as well as individual software and equipment vendors, to define TLVs that advertise additional information to Neighbor DRNI System. The Organization-Specific TLV structure shall be as shown in Figure 9-18 and further described in the field definitions after the figure.

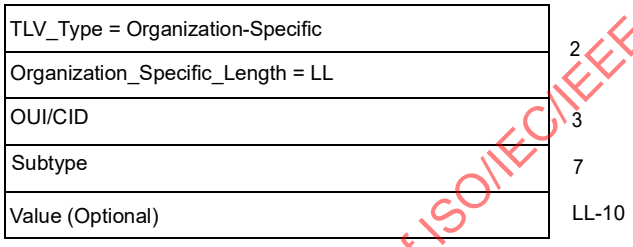



**Figure 9-18—Organization-Specific TLV**

a) *TLV_Type = Organization-Specific TLV.* Value is shown in Table 9-4.
b) *Organization_Specific_Length.* This field indicates the combined length (in octets) of the OUI/CID, Subtype, and Value fields of the TLV.
c) *OUI/CID.* This field contains the 3-byte OUI or CID, obtainable from the IEEE.
d) *Subtype.* This field contains a Subtype value, so that an additional OUI/CID will not be required if more Organization-Specific TLVs are required by an owner of an OUI/CID.
e) *Value.* This field contains the information that is communicated to a Neighbor DRNI System.

The following restrictions apply:

f) Information transmitted in an Organization-Specific TLV shall not require the recipient to violate any requirement in this standard.
g) Information transmitted in one Organization-Specific TLV shall not be used to provide a means for sending messages that are larger than would fit within a single DRCPDU.

### 9.6.3 DRCP state machine overview

Figure 9-19 is not itself a state machine but provides an overview of the DRCP state machines and the variables that facilitate their operation and interactions. Each DRCP state machine is represented as a box with one section containing a list of the primary variables controlled by that machine and another section containing a list of the functions invoked by that machine. Communication between machines is represented by arrows labeled with the variable being communicated. An arrow with a solid (black) arrowhead indicates that the variable is not modified by the destination machine. An arrow with a hollow (white) arrowhead indicates that the variable can be modified by the destination machine.

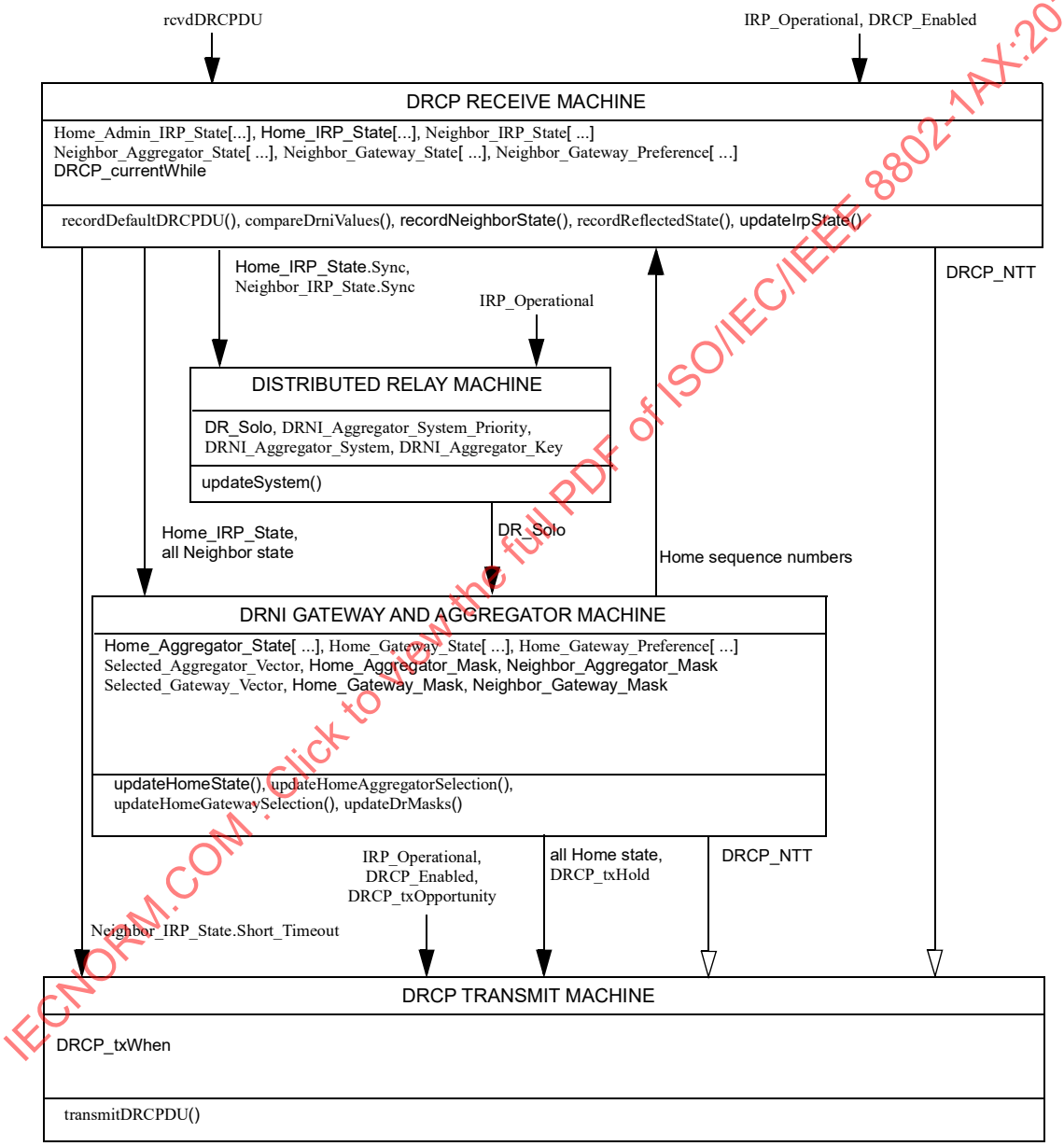



**Figure 9-19—DRCP state machine overview**

The state machines are as follows:

a) *DRCP Receive machine* (9.6.10). This state machine receives DRCPDUs from the Neighbor DRNI System on this IRP, records the information contained, and times it out using either Short Timeouts or Long Timeouts, according to the setting of DRCP_Timeout. It evaluates the incoming information from the Neighbor DRNI System to determine whether the Home and Neighbor have both agreed upon the protocol information exchanged to the extent that the Home DRNI System can now be safely used, either with a Neighbor DRNI System or as an individual DRNI System. If not, it asserts DRCP_NTT in order to transmit fresh protocol information to the Neighbor DRNI System. If the protocol information from the Neighbor DRNI System times out, the DRCP Receive machine installs default parameter values for use by the other state machines.
b) *Distributed Relay machine* (9.6.11). This state machine is responsible for establishing whether the Home and Neighbor DRNI Gateways can form a DRNI and to update the operational key of the Aggregator and all Aggregation Links accordingly.
c) *DRNI Gateway and Aggregator machine* (9.6.12). These state machines are responsible for configuring the Gateway Conversation IDs that are allowed to pass through this DRNI Gateway's DRNI Gateway Port and the Port Conversation IDs that are allowed to be distributed through this DRNI Gateway's Aggregator.
d) *DRCP Transmit machine* (9.6.13). This state machine handles the transmission of DRCPDUs, both on demand from the other state machines and on a periodic basis.

### 9.6.4 Constants

All timers specified in this subclause have an implementation tolerance of ± 250 ms.

Drcp_Fast_Periodic_Time
: The number of seconds between periodic transmissions using Short Timeouts.
Data Type: Integer.
Value: 1

Drcp_Slow_Periodic_Time
: The number of seconds between periodic transmissions using Long Timeouts.
Data Type: Integer.
Value: 30

Drcp_Short_Timeout_Time
: The number of seconds before invalidating received DRCPDU information when using Short Timeouts (3 × Drcp_Fast_Periodic_Time).
Data Type: Integer.
Value: 3

Drcp_Long_Timeout_Time
: The number of seconds before invalidating received DRCPDU information when using Long Timeouts (3 × Drcp_Slow_Periodic_Time).
Data Type: Integer.
Value: 90

NOTE—Timing out DRCPDU exchanges is the method of last resort for detecting IRC failures. The MAC_Operational status generated by the link hardware is the first choice for link failure detection. Clause 18 of IEEE Std 802.1Q-2018 provides a method for detecting link failures (also indicated via the MAC_Operational status) when hardware means are not applicable.

### 9.6.5 Variables associated with the DRNI Gateway

DRNI_DA
: The address to be used as the destination address for control frames generated in the DRNI Gateway for transmission on the IRP (9.6.1.1).

Data Type: 6 octets.
Administratively assigned.
Alias of aDrniProtocolDA (7.4.1.1.12).

Gateway_Enabled
Provides administrative control over the DRNI Gateway Port interface, and is made available to the Distributed Relay Client via the "MAC_Enabled" status parameter of the ISS. When FALSE ("down"), this DRNI Gateway Port is not selected for any Gateway Conversation IDs, and the operational state of the DRNI Gateway Port is "down" (Gateway_Operational is FALSE). When TRUE ("up"), the operational state of the DRNI Gateway Port is determined by the operational state of the Aggregation Port and the IRP.
Data Type: Boolean.
Administratively assigned.
Alias of aDrniGatewayAdminState (7.4.1.1.9).

Gateway_Operational
Indicates the operational state of the DRNI Gateway Port, and is made available to the Distributed Relay Client via the "MAC_Operational" status parameter of the ISS. When TRUE, the DRNI Gateway Port is "up" and is available for use by the Distributed Relay Client; when FALSE, the DRNI Gateway Port is "down" and is not available for use by the Distributed Relay Client.
Data Type: Boolean.
TRUE when Gateway_Enabled is TRUE and either of the following conditions are met:
1) Aggregator_Operational is TRUE, or
2) enableIrcData is TRUE and the Neighbor_Aggregator_State.Active_LAG_Links list is not empty.
Alias of aDrniGatewayOperState (7.4.1.1.10).

DRNI_Aggregator_System_Priority
The Aggregator System Priority value to be used by the Aggregator supporting this DRNI Gateway (and the Aggregation Ports assigned to this DRNI Gateway) when paired with a neighbor via the IRC. The default value is zero.
Data Type: Integer.
Administratively assigned.
Alias of aDrniAggregatorSystemPriority (7.4.1.1.14).

DRNI_Aggregator_System
The Aggregator System value to be used by the Aggregator supporting this DRNI Gateway (and the Aggregation Ports assigned to this DRNI Gateway) when paired with a neighbor DRNI System via the IRC. The default value is zero. The use and uniqueness requirements for this variable are described in 9.5.3.
Data Type: 6 octets.
Administratively assigned.
Alias of aDrniAggregatorSystem (7.4.1.1.13).

DRNI_Aggregator_Key
The Aggregator Key value to be used by the Aggregator supporting this DRNI Gateway (and the Aggregation Ports assigned to this DRNI Gateway) when paired with a neighbor DRNI System via the IRC. The default value is the Aggregator's Actor_Admin_Aggregator_Key value.
Data Type: Integer.
Administratively assigned.
Alias of aDrniAggregatorKey (7.4.1.1.15).

Home_Admin_Client_Gateway_Control
Administrative flag that, when TRUE, allows the Distributed Relay Client to determine whether to forward frames through the DRNI Gateway Port (see 9.5.3.5). The default value is

FALSE. A change to this variable, or a change to the forwarding state of the Distributed Relay Client port while this variable is TRUE, causes newHomeInfo to be set to TRUE.
Data Type: Boolean.
Alias of aDrniHomeDRClientGatewayControl (7.4.1.1.17).

Home_Admin_CSCD_Gateway_Control
Administrative flag that, when TRUE, allows the DRNI Gateway Port selection to be based on the CSCD parameters that drive the Aggregator Port selection. The default value is FALSE. A change to this variable causes newHomeInfo to be set to TRUE.
Data Type: Boolean.
Alias of aDrniHomeCscdGatewayControl (7.4.1.1.16).

Home_Admin_Gateway_Preference_Mask
A Boolean vector, indexed by Gateway Conversation ID, that indicates whether this DRNI System contains the preferred DRNI Gateway Port for the Gateway Conversation ID (TRUE = this DRNI System contains the preferred DRNI Gateway Port). The default value is TRUE for all Gateway Conversation IDs. A change to this variable causes newHomeInfo to be set to TRUE.
Data Type: sequence of Boolean values, indexed by Gateway Conversation ID.
Alias of aDrniHomeGatewayPreferenceMask (7.4.1.1.19).

Home_Admin_Gateway_Enable_Mask
A Boolean vector, indexed by Gateway Conversation ID, that indicates whether the Gateway Conversation ID has been enabled to pass through the Home DRNI Gateway Port by network administration or by a network control protocol (FALSE = blocked). The default value is TRUE for all Gateway Conversation IDs. A change to this variable causes newHomeInfo to be set to TRUE.
Data Type: sequence of Boolean values, indexed by Gateway Conversation ID.
Alias of aDrniHomeGatewayEnableMask (7.4.1.1.18).

Home_Admin_Gateway_Algorithm
The gateway algorithm administratively assigned for use by this DRNI Gateway to assign frames to Gateway Conversation IDs. The default value is “Unspecified” (Table 8-1). A change to this variable causes newHomeInfo to be set to TRUE.
Data Type: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).
Alias of aDrniHomeGatewayAlgorithm (7.4.1.1.6).

Home_Admin_Gateway_Conv_Service_Map
An array, indexed by Gateway Conversation ID, of entries, where each entry is a set of Service IDs (8.2) that map to that Conversation ID. Any given Service ID value can appear in at most one entry in the array. A Service ID value not contained in any entry does not map to any Conversation ID, and frames associated with that Service ID value are discarded. When modified, the Home_Gateway_Conv_Service_Digest is recalculated. A change to this variable when the most significant bit of the fourth octet of Home_Admin_Gateway_Algorithm is 1 causes newHomeInfo to be set to TRUE.
Data Type: Array, indexed by Conversation ID, of lists of Service IDs (8.2).
Administratively assigned.
Alias of aDrniHomeAdminGatewayConvServiceMap (7.4.1.1.20).

Home_Gateway_Conv_Service_Digest
A digest of Home_Admin_Gateway_Conv_Service_Map for exchange with the Neighbor DRNI System. The digest is a 16-octet MD5 fingerprint (see IETF RFC 1321). To calculate the digest, Home_Admin_Gateway_Conv_Service_Map is considered to contain 4096 consecutive elements, where each element contains a list of zero or more Service IDs (8.2), encoded as binary numbers in increasing order, followed by the Gateway Conversation ID to which they are mapped. The first element contains the Service ID(s) assigned to Gateway

Conversation ID 0, the second element contains the Service ID(s) assigned to Gateway Conversation ID 1, the third element contains the Service ID(s) assigned to Gateway Conversation ID 2, and so on, with the last element containing the Service ID(s) assigned to Gateway Conversation ID 4095.
Data Type: MD5 digest.
Alias of aDrniHomeGatewayConvServiceDigest (7.4.1.1.20).

Home_Aggregator_State
The Aggregator State of this DRNI Gateway, structured as an array with each entry consisting of the following fields as defined in 9.6.2.4:

Aggregator_Sequence_Number
A 32-bit sequence number, initialized to zero and incremented whenever any other field in the Aggregator State is updated by updateHomeState().

Aggregator_Port_Algorithm
The port algorithm used by the DRNI Gateway to assign frames to Port Conversation IDs, as determined by the value of the Aggregator's Actor_Port_Algorithm.

Aggregator_Conv_Service_Digest
The MD5 digest of the Aggregator's Admin_Conv_Service_Map.

Aggregator_Conv_Link_Digest
The MD5 digest of the Aggregator's Admin_Conv_Link_Map.

Partner_System_Priority
The value of the Aggregator's Partner_System_Priority.

Partner_System
The value of the Aggregator's Partner_System.

Partner_Oper_Aggregator_Key
The value of the Aggregator's Partner_Oper_Aggregator_Key.

Aggregator_CSCD_State
The value of the CSCD_State bits, as defined in 9.6.2.4.

Active_LAG_Links
The value of the Aggregator's Active_LAG_Links.

Home_Gateway_State
The Gateway State of this DRNI Gateway, structured as an array with each entry consisting of the following fields as defined in 9.6.2.5:

Gateway_Sequence_Number
A 32-bit sequence number, initialized to zero and incremented whenever any other field in the Gateway State is updated by updateHomeState().

Gateway_Algorithm
The gateway algorithm used by the DRNI Gateway to assign frames to Gateway Conversation IDs, as determined from the value of Home_Admin_Gateway_Algorithm.

Gateway_Conv_Service_Digest
If the most significant bit of the fourth octet of the Gateway_Algorithm is 1, this field contains the value of Home_Gateway_Conv_Service_Digest; otherwise, this field contains all zeros.

Gateway_Available_Mask
A sequence of Boolean values, indexed by Gateway Conversation ID, that indicates whether frames mapping to that Gateway Conversation ID could be forwarded through the DRNI Gateway Port.

Home_Gateway_Preference
: The Gateway Preference of this DRNI Gateway, structured as an array with each entry consisting of the following fields as defined in 9.6.2.5:

  Gateway_Preference_Sequence_Number
  : A 32-bit sequence number, initialized to zero and incremented whenever the Gateway_Preference_Vector is updated by updateHomeState().

  Gateway_Preference_Mask
  : The value of Home_Admin_Gateway_Preference_Mask.

Selected_Gateway_Vector
: Operational vector listing which DRNI Gateway's DRNI Gateway Port (if any) is passing each Gateway Conversation ID based on this DRNI Gateway's view of the operational state of the DRNI.
Data Type: sequence of 2 bit values ('00' for none, '01' for Home, and '10' for Neighbor), indexed by Gateway Conversation ID.

Selected_Aggregator_Vector
: Operational vector listing which DRNI Gateway's Aggregator Port (if any) is used to distribute each Port Conversation ID based on this DRNI Gateway's view of the operational state of the DRNI.
Data Type: sequence of 2 bit values ('00' for none, '01' for Home, and '10' for Neighbor), indexed by Port Conversation ID.

Gateway_Sync_Mask
: A Boolean vector, indexed by Gateway Conversation ID, that indicates whether the Gateway selection has been synchronized with the Neighbor. This variable is initialized by the Distributed Relay machine and maintained by the Distributed Relay machine and the DRNI Gateway and Aggregator machine.
Data Type: sequence of Boolean values, indexed by Gateway Conversation ID.

Home_Admin_IRP_State
: The administrative values of the state parameters for this IRP. This consists of the following set of variables, as described in 9.6.2.3:

  Reserved1
  Reserved2
  Short_Timeout
  Sync
  IRC_Data
  DRNI
  Defaulted
  Expired

When management changes the Short_Timeout or IRC_Data values in this variable, the new values are immediately copied to Home_IRP_State, DRCP_NTT is set to TRUE, and enableIrcData is reevaluated.
Data Type: 1 octet.
Alias of aDrniHomeAdminIrpState (7.4.1.1.24).

NOTE—The Sync, DRNI, Defaulted, and Expired variables of the Home_Admin_IRP_State are not used. They are included in the variable definition to maintain consistency with other IRP_State variables and to avoid confusion of the position of the bits within the field.

Home_IRP_State
: The operational values of the state parameters for this IRP. This consists of the following set of variables, as described in 9.6.2.3:

  Reserved1
  Reserved2
  Short_Timeout

Sync
IRC_Data
DRNI
Defaulted
Expired

Data Type: 1 octet.
Alias of aDrniHomeOperIrpState (7.4.1.1.25).

Neighbor_IRP_State
The operational value of the state parameters for the Neighbor's IRP. This consists of the following set of variables, as described in 9.6.2.3:

Reserved1
Reserved2
Short_Timeout
Sync
IRC_Data
DRNI
Defaulted
Expired

Data Type: 1 octet.
Alias of aDrniNeighborOperIrpState (7.4.1.1.26).

Neighbor_System_Priority
The last received System Priority of the Neighbor DRNI System on this IRP.
Data Type: Integer.
Alias of aDrniNeighborSystemPriority (7.4.1.1.30).

Neighbor_System
The last received MAC address component of System Identifier of the Neighbor DRNI System on this IRP.
Data Type: 6 octets.
Alias of aDrniNeighborSystem (7.4.1.1.29).

Neighbor_DRNI_Key
The last received DRNI key of the Neighbor DRNI System on this IRP.
Data Type: Integer.
Alias of aDrniNeighborDrniKey (7.4.1.1.31).

Neighbor_Aggregator_State
The values from the Home Aggregator State TLV (9.6.2.4) last received from the Neighbor DRNI System on this IRP. This consists of the following fields:

Aggregator_Sequence_Number
Aggregator_Port_Algorithm
Aggregator_Conv_Service_Digest
Aggregator_Conv_Link_Digest
Partner_System_Priority
Partner_System
Partner_Oper_Aggregator_Key
Aggregator_CSCD_State
Active_LAG_Links

Neighbor_Gateway_State
The values from the Gateway State TLV (9.6.2.5) last received from the Neighbor DRNI System on this IRP. This consists of the following fields:

Gateway_Sequence_Number
Gateway_Algorithm

Gateway_Conv_Service_Digest
Gateway_Available_Mask

Neighbor_Gateway_Preference
The values from the Gateway Preference TLV (9.6.2.6) last received from the Neighbor DRNI System on this IRP. This consists of the following fields:
Gateway_Preference_Sequence_Number
Gateway_Preference_Mask

Reflected_Aggregator_Sequence_Number
The Neighbor's view of this DRNI Gateway's Aggregator sequence number recorded from the last received DRCPDU from the Neighbor DRNI System on this IRP.

Reflected_Gateway_Sequence_Number
The Neighbor's view of this DRNI Gateway's Gateway sequence number recorded from the last received DRCPDU from the Neighbor DRNI System on this IRP.

Reflected_Gateway_Preference_Sequence_Number
The Neighbor's view of this DRNI Gateway's Gateway Preference sequence number recorded from the last received DRCPDU from the Neighbor DRNI System on this IRP.

### 9.6.6 Variables used for managing the operation of the state machines

BEGIN
This variable indicates the initialization (or reinitialization) of the DRCP protocol entity. It is set to TRUE when the System is initialized or reinitialized and is set to FALSE when (re)initialization has completed.
Data Type: Boolean.

Differ_DRNI
A Boolean indicating that the most recently received DRCPDU on this IRP is associated with a different DRNI.
Data Type: Boolean.

DRCP_Enabled
This variable indicates that the associated IRP is operating the DRCP. If the link is not a point-to-point link, the value of DRCP_Enabled shall be FALSE; otherwise, the value of DRCP_Enabled shall be TRUE.
Data Type: Boolean.

DRCP_NTT
Need To Transmit flag.
Data Type: Boolean.
TRUE indicates that there is new protocol information to be transmitted on this IRP or that the Neighbor DRNI System needs to be reminded of the old information.
FALSE otherwise.

DRCP_Timeout
The timeout value for the DRCP Receive machine timer, set to either Drcp_Short_Timeout_Time or Drcp_Long_Timeout_Time by the DRCP Receive machine according to the most recently received value of Neighbor_IRP_State.Short_Timeout.
Data Type: Integer.

DRCP_txHold
A Boolean signal that, when TRUE, inhibits the transmission of new Home state information, or the acknowledgment of new Neighbor state information, until that information has been processed and applied to the DRNI Gateway forwarding.
Data Type: Boolean.

DRCP_txOpportunity
A Boolean signal from the System that is TRUE when DRCPDU transmission is permitted and FALSE when transmission of DRCPDUs is inhibited due to transmit resource contention.
Data Type: Boolean.

IRP_Operational
An alias for the MAC_Operational status parameter of the Intra-Relay Port ISS.
Data Type: Boolean.
TRUE if the underlying service is operable.
FALSE otherwise.

NOTE—The means by which the value of the IRP_Operational variable is generated by the underlying service [e.g., by a MAC or Maintenance End Point (MEP) (Clause 18 of IEEE Std 802.1Q)] is implementation-dependent.

newHomeInfo
This variable indicates that the values for the Home_Gateway_State, Home_Gateway_Preference, and/or Home_Aggregator_State variables need to be updated in response to local administrative changes or information received from the DRCP Neighbor or LACP Partner.
Data Type: Boolean.

newNeighborState
This variable indicates that the DRCP Receive machine has received new values for the Neighbor_System_Priority, Neighbor_System, Neighbor_Gateway_State, Neighbor_Gateway_Preference, and/or Neighbor_Aggregator_State variables.
Data Type: Boolean.

newReflectedState
This variable indicates that the DRCP Receive machine has received new values for the Reflected_Gateway_Sequence_Number, Reflected_Gateway_Preference_Sequence_Number, and/or Reflected_Aggregator_Sequence_Number variables.
Data Type: Boolean.

differPartner
A Boolean that, when set to TRUE by the DRCP Receive machine, indicates the Aggregator supporting this DRNI Gateway has a different LACP partner from the Aggregator supporting the neighbor DRNI Gateway.
Data Type: Boolean.

newOperSystem
A Boolean that, when set to TRUE by the updateSystem function, indicates one or more of the values of the Aggregator's Actor_Oper_System_Priority, Actor_Oper_System, or Actor_Oper_Aggregator_Key variable have changed.
Data Type: Boolean.

DR_Solo
A Boolean that, when TRUE, indicates this DRNI Gateway is not paired with a DRNI Gateway in another DRNI System.
Data Type: Boolean.

GatewayFollowsAggregator
This variable, when TRUE, indicates that from the Home DRNI Gateway's view of the DRNI, the DRNI Gateway Port is in the same DRNI System as the Aggregator Port for all Conversation IDs.
Data Type: Boolean.

NeighborHasLowestID
This variable, when TRUE, indicates that the concatenation of the Neighbor_System_Priority and Neighbor_System is numerically lower than the Home System Identifier (9.5.3.1).
Data Type: Boolean.

### 9.6.7 Functions

recordDefaultDRCPDU
Home_IRP_State.Short_Timeout = Home_Admin_IRP_State.Short_Timeout;
Home_IRP_State.Sync = FALSE;
Home_IRP_State.IRC_Data = Home_Admin_IRP_State.IRC_Data;
Home_IRP_State.Defaulted = TRUE;
Home_IRP_State.Expired = FALSE;

Neighbor_System_Priority = 0;
Neighbor_System = 0;

All values in Neighbor_IRP_State, Neighbor_Gateway_State, Neighbor_Gateway_Preference, and Neighbor_Aggregator_State are set to null (i.e., all numerical values set to 0, all Boolean values set to FALSE, and all lists set to empty lists);

DRCP_Timeout = Drcp_Short_Timeout_Time;
Differ_DRNI = FALSE;
differPartner = FALSE;
newHomeInfo = TRUE;
newNeighborState = TRUE;
newReflectedState = TRUE.

compareDrniValues
This function verifies that the Neighbor DRNI Gateway has the same DRNI Aggregator identification values as the Home DRNI Gateway and that the Neighbor has differing System identification values and sets the Differ_DRNI accordingly.

The parameter values identifying the DRNI Aggregator and transmitting System carried in the received DRCPDU are compared with the corresponding parameter values for this DRNI Aggregator and System. If

*Version Number* is less than 2, or

*DRNI_Aggregator_System_Priority* is not equal to DRNI_Aggregator_System_Priority, or

*DRNI_Aggregator_System* is not equal to DRNI_Aggregator_System, or

*DRNI_Aggregator_System* is nonzero and *DRNI_Aggregator_Key* is not equal to DRNI_Aggregator_Key, or

*Home_System* is equal to Actor_Admin_System, then
The variable Differ_DRNI is set to TRUE;

Otherwise:
The variable Differ_DRNI is set to FALSE.

recordReflectedState
This function records sequence numbers echoed by the Neighbor in acknowledgment of state information about the Home DRNI Gateway sent in DRCPDUs from this DRNI System and determines whether the Neighbor's view of the Home DRNI Gateway's state is current.

The *Neighbor_Gateway_Sequence_Number*, *Neighbor_Gateway_Preference_Sequence_Number*, and *Neighbor_Aggregator_Sequence_Number* in the received DRCPDU are compared to the Reflected_Gateway_Sequence_Number, Reflected_Gateway_Preference_Sequence_Number, and Reflected_Aggregator_Sequence_Number variables, respectively. If any of the sequence numbers in the DRCPDU are less, then

DRCP_NTT is set to TRUE so that the DRCP Transmit machine sends updated state to the Neighbor.

For any of the sequence numbers in the DRCPDU that are greater,

newReflectedState is set to TRUE, and

The sequence number(s) in the DRCPDU is copied to the Reflected_Gateway_Sequence_Number, Reflected_Gateway_Preference_Sequence_Number, and/or Reflected_Aggregator_Sequence_Number variable(s) as appropriate.

The *Neighbor_Gateway_Sequence_Number*, *Neighbor_Gateway_Preference_Sequence_Number*, and *Neighbor_Aggregator_Sequence_Number* in the received DRCPDU are compared to the sequence numbers in the Home_Gateway_State, Home_Gateway_Preference, and Home_Aggregator_State variables, respectively. If any of the sequence numbers in the DRCPDU are greater, then newHomeInfo is set to TRUE.

recordNeighborState

This function records sequence numbers and state information about the Neighbor DRNI Gateway provided in the received DRCPDU and sets flags for further processing of new information as appropriate.

This function compares the parameter values carried in the DRCPDU that identify the Neighbor DRNI System and key (*Home_System_Priority, Home_System,* and *DRNI_Aggregator_Key*) to those previously stored for the Neighbor (Neighbor_System_Priority, Neighbor_System, and Neighbor_DRNI_Key). If one or more of the comparisons show that the value(s) received in the DRCPDU differ from the currently stored values, then newNeighborState, DRCP_txHold, and DRCP_NTT are set to TRUE, and the stored value(s) are updated with the corresponding values received in the DRCPDU.

The *Home_Gateway_Sequence_Number*, *Home_Gateway_Preference_Sequence_Number,* and *Home_Aggregator_Sequence_Number* in the *DRNI_State_TLV* of the received DRCPDU are compared to the sequence numbers in the Neighbor_Gateway_State, Neighbor_Gateway_Preference, and Neighbor_Aggregator_State variables, respectively. If any sequence numbers do not match, then DRCP_NTT is set to TRUE. If any sequence number in the DRCPDU is greater and the DRCPDU includes a corresponding state TLV (*Home_Gateway_State_TLV, Home_Gateway_Preference_TLV*, or *Home_Aggregator_State_TLV*) with the same sequence number as in the *DRNI_State_TLV* of the received DRCPDU, then all fields of the corresponding TLV are copied to the corresponding variable for the Neighbor (Neighbor_Gateway_State, Neighbor_Gateway_Preference, or Neighbor_Aggregator_State), and newNeighborState, DRCP_txHold, and DRCP_NTT are set to TRUE.

If the Neighbor_Aggregator_State.Active_LAG_Links is not empty, and the Home_Aggregator_State.Active_LAG_Links is not empty, and any of the Partner_System_Priority, Partner_System, or Partner_Oper_Aggregator_Key values in the

Neighbor_Aggregator_State differ from the corresponding values in the Home_Aggregator_State, then the differPartner and newHomeInfo variables are set to TRUE.

updateIrpState

This function updates the Sync bits in the Home_IRP_State and Neighbor_IRP_State variables that determine the DRNI Gateways are ready to form a DRNI.

Home_IRP_State.Expired and Home_IRP_State.Defaulted are set to FALSE, and Home_IRP_State.Sync is set to TRUE.

The *Short_Timeout*, *IRC_Data*, *DRNI*, *Expired*, and *Defaulted* bits of the *Home_IRP_State* in the received DRCPDU are compared to the corresponding bits of the Neighbor_IRP_State variable. If one or more of the comparisons show that the value(s) received in the DRCPDU differ from the currently stored values, then DRCP_NTT is set to TRUE, and the stored value(s) are updated with the corresponding values received in the DRCPDU. If the newly updated value of Neighbor_IRP_State.Short_Timeout is TRUE, then DRCP_Timeout is set to Drcp_Short_Timeout_Time; otherwise, DRCP_Timeout is set to Drcp_Long_Timeout_Time.

The Neighbor's view of this DRNI Gateway's identification values in the received DRCPDU (*Neighbor_System_Priority* and *Neighbor_System*) are compared to this DRNI System's identification values (Actor_Admin_System_Priority and Actor_Admin_System). If they differ, or if Home_IRP_State.Sync is TRUE and the Home System Identifier is numerically lower than the Neighbor System Identifier and the *DRNI_Aggregator_Key* value in the received DRCPDU differs from the DRNI_Aggregator_Key variable, then DRCP_NTT is set to TRUE, and Neighbor_IRP_State.Sync is set to FALSE (because the Neighbor cannot be synchronized if its view of this DRNI System's identification is obsolete). Otherwise, Neighbor_IRP_State.Sync is set to the *Home_IRP_State.Sync* value received in the DRCPDU. If this results in a change to Neighbor_IRP_State.Sync, then DRCP_NTT is set to TRUE.

If the *Neighbor_IRP_State* carried in the receive DRCPDU differs from the Home_IRP_State value, then DRCP_NTT is set to TRUE.

updateSystem

This function updates the operational System Identifier of the Aggregator and Aggregation Ports according to whether this DRNI System is in a DRNI with another DRNI System.

If DR_Solo is TRUE, then

The Gateway_Sync_Mask is set to TRUE for all Gateway Conversation IDs, and

The Actor_Oper_System_Priority, Actor_Oper_System, and Actor_Oper_Aggregator_Key values for the Aggregator supporting this DRNI Gateway are set equal to the Aggregator's Actor_Admin_System_Priority, Actor_Admin_System, and Actor_Admin_Aggregator_Key, respectively.

Otherwise:

The Gateway_Sync_Mask is set to FALSE for all Gateway Conversation IDs, and

The Actor_Oper_System_Priority, Actor_Oper_System, and Actor_Oper_Aggregator_Key values for the Aggregator supporting this DRNI Gateway are set equal to DRNI_Aggregator_System_Priority, DRNI_Aggregator_System, and DRNI_Aggregator_Key, respectively, if the DRNI_Aggregator_System value is nonzero, or are set equal to the Actor_Admin_System_Priority, Actor_Admin_System, and DRNI_Aggregator_Key if the DRNI_Aggregator_System value is zero and the Home System Identifier is numerically lower than the Neighbor System Identifier, or are set equal to the Neighbor_System_Priority, Neighbor_System, and Neighbor_DRNI_Key if

the DRNI_Aggregator_System value is zero and the Neighbor System Identifier is numerically lower than the Home System Identifier.

If this results in a change to any of the Aggregator's operational values, then

The Actor_Oper_Port_System_Priority, Actor_Oper_Port_System, and Actor_Oper_Port_Key values of each Aggregation Port assigned to this DRNI Gateway are set to the Aggregator's Actor_Oper_System_Priority, Actor_Oper_System, and Actor_Oper_Aggregator_Key values, respectively, and

The newOperSystem and newHomeInfo variables are set to TRUE.

resetHomeState

All values in Home_Aggregator_State, Home_Gateway_State, and Home_Gateway_Preference are set to null (i.e., all numerical values set to 0, all Boolean values set to FALSE, and all lists set to empty lists). DRCP_NTT and newOperSystemare set to FALSE.

updateHomeState

This function updates the Home_Aggregator_State, Home_Gateway_State, and Home_Gateway_Preference variables in response to changes resulting from administrative action or the operation of other protocols (e.g., LACP).

If the Reflected_Aggregator_Sequence_Number, Reflected_Gateway_Sequence_Number, or Reflected_Gateway_Preference_Sequence_Number is greater than the sequence number in the Home_Aggregator_State, Home_Gateway_State, or Home_Gateway_Preference, respectively, then the sequence number in the Home state variable is set to one greater than the reflected sequence number.

If newOperSystem is TRUE, or if DR_Solo is FALSE and differPartner is TRUE and NeighborHasLowestID is TRUE, then

The Selected variable for each Aggregation Port is set to UNSELECTED, and

The Active_LAG_Links variable of the Aggregator is cleared, and

The changeAggregationLinks variable is set to TRUE, and

The newOperSystem and differPartner are set to FALSE.

If any of the fields in the Home_Aggregator_State differ from the current value of the variables from which they are derived (as described in the definition of Home_Aggregator_State in 9.6.5), then

DRCP_txHold is set to TRUE, and

If there is a change to the Aggregator_CSCD_State.CSCD_Gateway_Control, or if there is a change to the Aggregator_Port_Algorithm, Aggregator_Conv_Service_Digest, or Aggregator_Conv_Link_Digest while the CSCD_Gateway_Control value is TRUE, then the Gateway_Sync_Mask is set to FALSE for all Gateway Conversation IDs, and

If there is a change to a Link_Number in the Active_LAG_Links while the CSCD_Gateway_Control value is TRUE, then the Gateway_Sync_Mask is set to FALSE for any Gateway Conversation ID where the Admin_Conv_Link_Map entry for that Conversation ID contains that Link_Number, and

The fields of the Home_Aggregator_State are updated as specified in 9.6.5, and

If the Aggregator_Sequence_Number is equal to the Aggregator_Sequence_Number most recently transmitted by the transmitDRCPDU function, then the Aggregator_Sequence_Number is incremented.

A new Gateway_Available_Mask is calculated, for each Gateway Conversation ID, by setting the bit in the vector to TRUE if and only if the corresponding bit of the Home_Admin_Gateway_Enable_Mask vector is TRUE, and Gateway_Enabled is TRUE, and either Home_Admin_Client_Gateway_Control is FALSE or the Distributed Relay Client allows forwarding of frames with that Gateway Conversation ID through the DRNI Gateway Port (see 9.5.3.5). If the new Gateway_Available_Mask differs from the value in the Home_Gateway_State, or the Home_Admin_Gateway_Algorithm differs from the Gateway_Algorithm value in the Home_Gateway_State, or the most significant bit of the fourth octet of the Home_Admin_Gateway_Algorithm is 1 and the Home_Gateway_Conv_Service_Digest differs from the Gateway_Conv_Service_Digest value in the Home_Gateway_State, then

DRCP_txHold is set to TRUE, and

The Gateway_Sync_Mask is set to FALSE for any Gateway Conversation ID where the new Gateway_Available_Mask differs from the value in the Home_Gateway_State, and

if the Home_Admin_Gateway_Algorithm differs from the Gateway_Algorithm value in the Home_Gateway_State, or the most significant bit of the fourth octet of the Home_Admin_Gateway_Algorithm is 1 and the Home_Gateway_Conv_Service_Digest differs from the Gateway_Conv_Service_Digest value in the Home_Gateway_State, then the Gateway_Sync_Mask is set to FALSE for all Gateway Conversation IDs, and

The fields of the Home_Gateway_State are updated as specified in 9.6.5, and

If the Gateway_Sequence_Number is equal to the Gateway_Sequence_Number most recently transmitted by the transmitDRCPDU function, then the Gateway_Sequence_Number is incremented.

If the Gateway_Preference_Mask in the Home_Gateway_Preference variable differs from the current value of Home_Admin_Gateway_Preference_Mask, then

DRCP_txHold is set to TRUE, and

The Gateway_Sync_Mask is set to FALSE for any Gateway Conversation ID where the Home_Gateway_Preference differs from the Home_Admin_Gateway_Preference_Mask, and

The Gateway_Preference_Mask of the Home_Gateway_Preference is updated to the current value of Home_Admin_Gateway_Preference_Mask, and

If the Gateway_Preference_Sequence_Number is equal to the Gateway_Preference_Sequence_Number most recently transmitted by the transmitDRCPDU function, then the Gateway_Preference_Sequence_Number is incremented.

If any of the Home state variables are updated, then DRCP_NTT is set to TRUE.

setDefaultDRNISystemParameters

This function initializes the Home and Neighbor Aggregator Port and DRNI Gateway Port selection variables and the DRNI Gateway frame forwarding masks as follows:

For each Port Conversation ID:

The Selected_Aggregator_Vector is set to '00';
The Home_Aggregator_Mask is set to FALSE;
The Neighbor_Aggregator_Mask is set to FALSE.

For each Gateway Conversation ID:
The Selected_Gateway_Vector is set to '00';
The Home_Gateway_Mask is set to FALSE;
The Neighbor_Gateway_Mask is set to FALSE.

updateHomeAggregatorSelection

This function uses this DRNI Gateway's view of the DRNI state and the Aggregator's Conversation-Sensitive Distribution configuration to generate the Selected_Aggregator_Vector that associates each Port Conversation ID to the DRNI Gateway containing the Aggregation Link to be used for distributing frames with that Port Conversation ID.

If DR_Solo is TRUE, or the Home_Aggregator_State.Aggregator_Port_Algorithm is Unspecified, then
The Selected_Aggregator_Vector is set to '01' (Home) for all Port Conversation IDs;

Otherwise:
For each Port Conversation ID, the function evaluates the list of Link_Numbers contained in the Aggregator's Admin_Conv_Link_Map and selects the first value that matches a value in the list of Active_LAG_Links in either the Home_Aggregator_State or the Neighbor_Aggregator_State variable. If the first matching Link_Number is in the Home_Aggregator_State, the entry in the Selected_Aggregator_Vector for that Port Conversation ID is set to '01'. If the first matching Link_Number is in the Neighbor_Aggregator_State, the entry in the Selected_Aggregator_Vector for that Port Conversation ID is set to '10'. If no matching value is found, or if the function encounters the value zero in the list from the Admin_Conv_Link_Map before a matching value is found, then the entry in the Selected_Aggregator_Vector for that Port Conversation ID is '00'.

updateHomeGatewaySelection

This function uses this DRNI Gateway's view of the DRNI state to generate the Selected_Gateway_Vector that associates each Gateway Conversation ID to the DRNI Gateway with the DRNI Gateway Port selected to forward frames mapping to that Gateway Conversation ID.

The local variable GatewayFollowsAggregator is TRUE when Aggregator_CSCD_State.CSCD_Gateway_Control is TRUE in both the Home_Aggregator_State and Neighbor_Aggregator_State variables, and the Aggregator_Port_Algorithm and Aggregator_Conv_Service_Digest values in the Home_Aggregator_State match the corresponding values in the Neighbor_Aggregator_State and also match the Gateway_Algorithm and Gateway_Conv_Service_Digest values in the Home_Gateway_State and Neighbor_Gateway_State variables. Otherwise, GatewayFollowsAggregator is FALSE.

For each Gateway Conversation ID, the value for the Selected_Gateway_Vector is determined as follows:

If DR_Solo is TRUE, then the Selected_Gateway_Vector is set to '01' (Home) if the Home_Gateway_State.Gateway_Available_Mask is TRUE and set to '00' if the Home_Gateway_State.Gateway_Available_Mask is FALSE;

Otherwise, if the Home_Gateway_State.Gateway_Algorithm is 'Unspecified', or the values for the Gateway_Algorithm values in the Neighbor_Gateway_State and Home_Gateway_State are not the same, or the most significant bit of the fourth octet of the Gateway_Algorithm is 1 and the Gateway_Conv_Service_Digest values in the Neighbor_Gateway_State and Home_Gateway_State are not the same, then

The Selected_Gateway_Vector is set to '10' (Neighbor) if the NeighborHasLowestID variable is TRUE, or set to '01' (Home) if the NeighborHasLowestID variable is FALSE and the Home_Gateway_State.Gateway_Available_Mask is TRUE, or set to '00' if the NeighborHasLowestID variable is FALSE and the Home_Gateway_State.Gateway_Available_Mask is FALSE;

Otherwise, if the Neighbor_Gateway_State.Gateway_Available_Mask value and the Home_Gateway_State.Gateway_Available_Mask value are both FALSE, then

The Selected_Gateway_Vector is set to '00';

Otherwise, if the Neighbor_Gateway_State.Gateway_Available_Mask value is TRUE and the Home_Gateway_State.Gateway_Available_Mask value is FALSE, then

The Selected_Gateway_Vector is set to '10' (Neighbor);

Otherwise, if the Neighbor_Gateway_State.Gateway_Available_Mask value is FALSE and the Home_Gateway_State.Gateway_Available_Mask value is TRUE, then

The Selected_Gateway_Vector is set to '01' (Home);

Otherwise, if the Neighbor_Gateway_State.Gateway_Available_Mask value and the Home_Gateway_State.Gateway_Available_Mask value are both TRUE and GatewayFollowsAggregator is TRUE, then

The Selected_Gateway_Vector is set to the Selected_Aggregator_Vector value;

Otherwise, if the Neighbor_Gateway_Preference.Gateway_Preference_Mask value is TRUE and the Home_Gateway_Preference.Gateway_Preference_Mask value is FALSE, then

The Selected_Gateway_Vector is set to '10' (Neighbor);

Otherwise, if the Neighbor_Gateway_Preference.Gateway_Preference_Mask value is FALSE and the Home_Gateway_Preference.Gateway_Preference_Mask value is TRUE, then

The Selected_Gateway_Vector is set to '01' (Home);

Otherwise:

The Selected_Gateway_Vector is set to '10' (Neighbor) if the NeighborHasLowestID variable is TRUE or set to '01' (Home) if the NeighborHasLowestID variable is FALSE.

updateDrMasks

This function creates the four Boolean masks used by the DRNI Gateway frame forwarding state machines (9.5.2).

If DR_Solo is TRUE or the sequence numbers in each of the Home_Aggregator_State, Home_Gateway_State, and Home_Gateway_Preference match the Reflected_Aggregator_Sequence_Number, Reflected_Gateway_Sequence_Number, and Reflected_Gateway_Preference_Sequence_Number, respectively, then

The Gateway_Sync_Mask value is set to TRUE for all Gateway Conversation IDs.

For each Gateway Conversation ID, if the Selected_Gateway_Vector does not equal '01' (Home), then

The Home_Gateway_Mask is set to FALSE;

Otherwise, if the Gateway_Sync_Mask for that Gateway Conversation ID is TRUE, then

The Home_Gateway_Mask is set to TRUE;

Otherwise:

The Home_Gateway_Mask remains unchanged.

For each Gateway Conversation ID, if the Selected_Gateway_Vector does not equal '10' (Neighbor), then
The Neighbor_Gateway_Mask is set to FALSE;
Otherwise, if the Gateway_Sync_Mask for that Gateway Conversation ID is TRUE, then
The Neighbor_Gateway_Mask is set to TRUE;
Otherwise:
The Neighbor_Gateway_Mask remains unchanged.

For each Port Conversation ID, the Home_Aggregator_Mask is TRUE if the value in the Selected_Aggregator_Vector is equal to the '01' (Home) and FALSE otherwise.

For each Port Conversation ID, the Neighbor_Aggregator_Mask is TRUE if the value in the Selected_Aggregator_Vector is equal to the '10' (Neighbor) and FALSE otherwise.

NOTE—In implementations, these masks are either to be updated atomically (that is, all frame forwarding paused while the masks are updated) or to be updated in a "break-before-make" manner (that is, first update all variables with the logical 'AND' of the new value and the old value, and then update all variables with the new value).

transmitDRCPDU
This function generates a DRCPDU, formatted as specified in 9.6.2, and issues it as a IrpCtrl:M_UNITDATA.Request(DRCPDU) service primitive to the DRCP Parser/Multiplexer (9.5.1). This function also saves the values of the Aggregator_Sequence_Number, Gateway_Sequence_Number, and Gateway_Preference_Sequence_Number transmitted in this DRCPDU.

### 9.6.8 Timers

DRCP_currentWhile
This timer is used by the DRCP Receive machine to detect whether received protocol information has expired. If Home_IRP_State.Short_Timeout is TRUE, the timer is started with the value Short_Timeout_Time; otherwise, it is started with the value Long_Timeout_Time (see 9.6.4).
DRCP_txWhen
This timer is used by the DRCP Transmit machine to generate periodic transmissions. It is started using the value Slow_Periodic_Time or Fast_Periodic_Time (see 9.6.4), depending on the value of Neighbor_IRP_State.Short_Timeout.

### 9.6.9 Messages

rcvdDRCPDU
Set to TRUE in response to a IrpCtrl:M_UNITDATA.indication(DRCPDU) service primitive generated by the DRCP Parser/Multiplexer as a result of the reception of a DRCPDU, formatted as defined in 9.6.2.3. The contents of the DRCPDU are available to the DRCP Receive machine until the DRCP Receive machine sets rcvdDRCPDU to FALSE.
Data Type: Boolean.

### 9.6.10 DRCP Receive machine

The DRCP Receive machine shall implement the function specified in Figure 9-20 with its associated parameters (9.6.4 through 9.6.9). There is one DRCP Receive machine for the IRP in a DRNI Gateway.

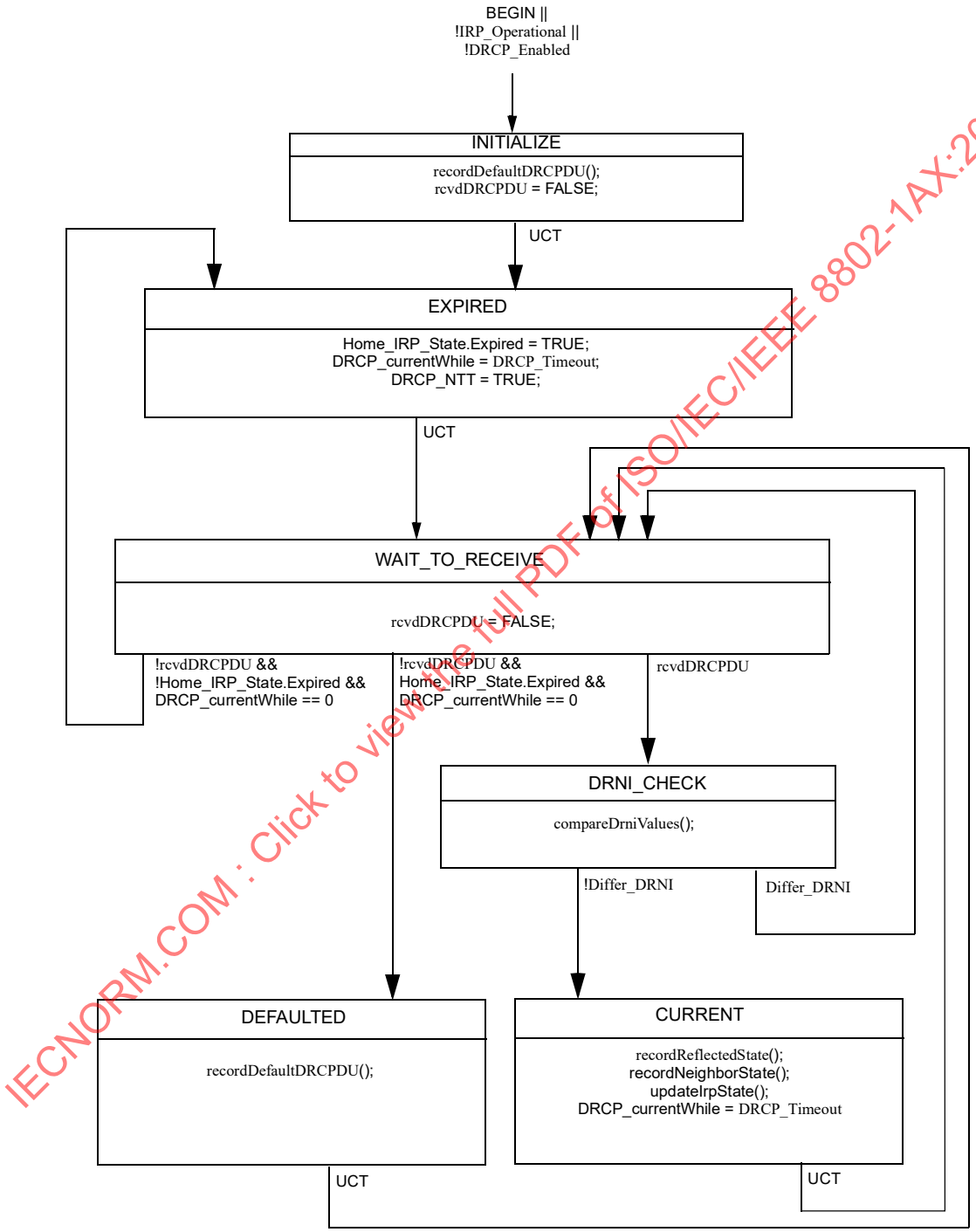



**Figure 9-20—DRCP Receive machine state diagram**

Upon initialization (BEGIN), or if the IRP becomes inoperable or DRCP operation is disabled, the state machine enters the INITIALIZE state. The recordDefaultDRCPDU function clears all variables containing information received from a previous Neighbor DRNI System. The Home_IRP_State.Sync bit is set to FALSE, which causes the Distributed Relay machine to put the DRNI Gateway in solo operation. The newHomeInfo and newNeighborState are set to TRUE, which triggers the DRNI Gateway and Aggregator machine to recompute the Gateway and Aggregator Masks that drive the DRNI Gateway frame forwarding.

Following initialization, when the conditions for the global entry into INITIALIZE are no longer met (i.e., BEGIN is not asserted, the IRP is operational, and DRCP is enabled), the state machine enters the EXPIRED state. The receive timer is started, and the state machine continues to the WAIT_TO_RECEIVE state.

When the state machine is in the WAIT_TO_RECEIVE state, receipt of a DRCPDU causes a transition to the DRNI_CHECK state. The compareDrniValues function checks if the DRCPDU is associated with this DRNI. If not, the state machine returns to the WAIT_TO_RECEIVE state. If the compareDrniValues function identifies the received DRCPDU as associated with this DRNI, the state machine enters the CURRENT state.

Upon entry to the CURRENT state, the recordNeighborState and recordReflectedState functions record the information contained in the DRCPDU regarding the Neighbor DRNI Gateway and regarding the Neighbor DRNI Gateway's view of the Home DRNI Gateway state. If new information is received, the newNeighborState and/or newReflectedState are set to TRUE, which triggers the DRNI Gateway and Aggregator machine to recompute the Gateway and Aggregator Masks that drive the DRNI Gateway frame forwarding. The updateIrpState function examines the information received in the DRCPDU to determine whether both the Home and Neighbor DRNI Gateways agree that their information is current and that they are prepared to form a DRNI. DRCP_NTT is set to TRUE if any of these functions detect that the Neighbor is using information about the Home DRNI Gateway that is obsolete. The DRCP_currentWhile timer is started with either Drcp_Short_Timeout_Time or Drcp_Long_Timeout_Time, depending upon the current operational value of Neighbor_IRP_State.Short_Timeout, and the state machine returns to the WAIT_TO_RECEIVE state.

If the timer expires before a DRCPDU associated with this DRNI is received, the state machine returns to the EXPIRED state or continues to the DEFAULTED state, depending on whether Home_IRP_State.Expired is already TRUE. In the DEFAULTED state, the recordDefaultDRCPDU function clears all variables containing information received from a previous Neighbor DRNI System. The Home_IRP_State.Sync bit is set to FALSE, which causes the Distributed Relay machine to put the DRNI Gateway in solo operation. The newHomeInfo and newNeighborState are set to TRUE, which triggers the DRNI Gateway and Aggregator machine to recompute the Gateway and Aggregator Masks that drive the DRNI Gateway frame forwarding. Then the state machine returns to the WAIT_TO_RECEIVE state.

In the process of executing the compareDrniValues function, a DRCP Receive machine compliant to this standard shall validate a received DRCPDU by verifying that the Version Number in a received DRCPDU is greater than or equal to 2 and may verify that the TLV_Length associated with any TLV_Type defined in this standard is at least the value specified in 9.6.2. Receipt of an invalid DRCPDU results in Differ_DRNI set to TRUE and a return to the WAIT_TO_RECEIVE state. A received DRCPDU is not considered invalid because it contains one or more TLVs with a TLV_Type not defined in this standard; the information in such TLVs is ignored. A received DRCPDU is not considered invalid because it contains one or more TLVs with a TLV_Length greater than defined in this standard for that TLV_Type; the excess information in such TLVs (beyond the length defined in this standard) is ignored.

### 9.6.11 Distributed Relay machine

The Distributed Relay machine shall implement the function specified in Figure 9-21 with its associated parameters (9.6.4 through 9.6.9).

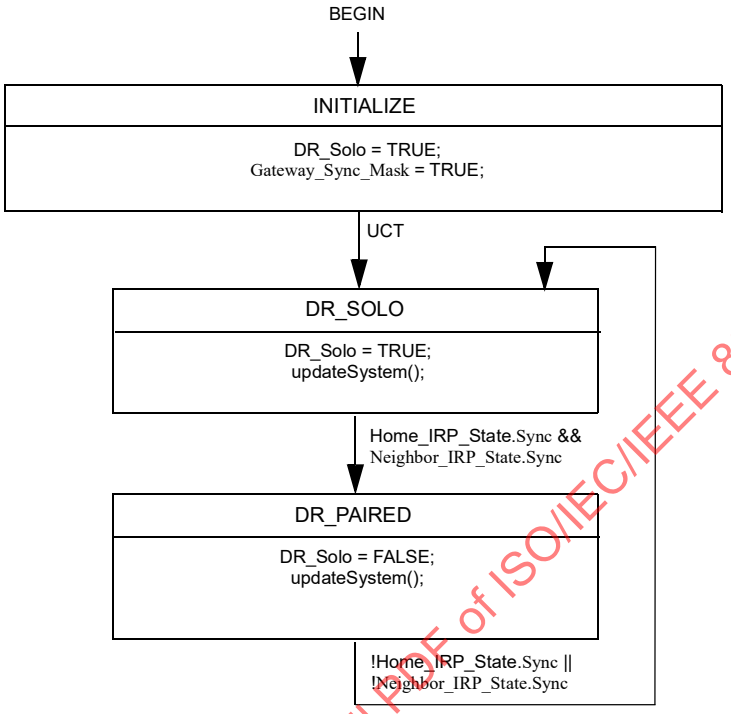



**Figure 9-21—Distributed Relay machine state diagram**

The Distributed Relay machine is responsible for determining whether this DRNI Gateway is connected via an IRC to another DRNI Gateway with compatible configuration to form a DRNI and for managing the operational Key value of the Aggregator accordingly.

On initialization, the DRNI Gateway is in the DR_SOLO state to indicate that the DRNI Gateway is operating solo (not having formed a DRNI with another DRNI System). The operational actor identifier and key values of the Aggregator are set to the administrative actor values.

When the DRCP Receive machine indicates that both the Home and Neighbor DRNI Gateways are synchronized (the IRP is operational and the connected DRNI Systems have compatible configurations), the state machine enters the DR_PAIRED state. If the DRNI_Aggregator_System value is nonzero, then the operational actor identifier and key value of the Aggregator are updated to use the DRNI_Aggregator_System_Priority, DRNI_Aggregator_System, and DRNI_Aggregator_Key values. If the DRNI_Aggregator_System value is zero and the Home System Identifier is numerically lower than the Neighbor System Identifier, then the Aggregator's Actor_Admin_System_Priority and Actor_Admin_System values, together with the DRNI_Aggregator_Key value, are used for the operational actor identifier and key. Otherwise, the Neighbor_System_Priority, Neighbor_System, and Neighbor_DRNI_Key values are used for the operational actor identifier and key.

If the DRCP Receive machine indicates that the Home and Neighbor DRNI Gateways are not synchronized (the configuration has changed to be incompatible or the IRP is no longer operational), the state machine returns to the DR_SOLO state and restores the Aggregator operational actor identifier and key values to the administrative actor value.

### 9.6.12 DRNI Gateway and Aggregator machine

The DRNI Gateway and Aggregator machine shall implement the function specified in Figure 9-22 with their associated parameters (9.6.4 through 9.6.9). This state machine is responsible for determining the Gateway Conversation IDs and the Port Conversation IDs that are allowed to pass through this DRNI Gateway's DRNI Gateway Port and Aggregator Port based on the current state of the DRNI and the agreed priority rules (the Gateway_Preference_Masks and the Aggregator's Admin_Conv_Link_Map).

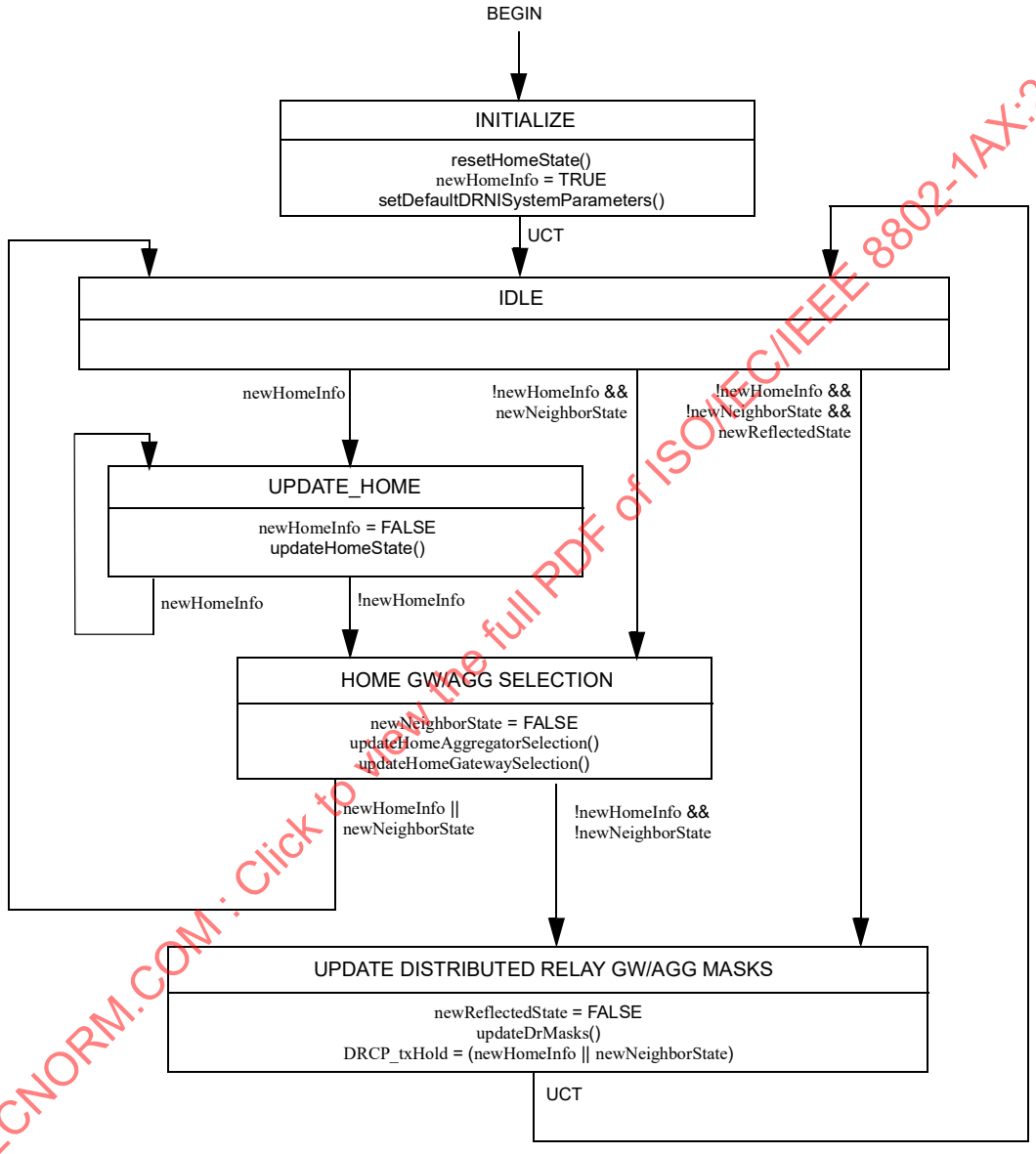



**Figure 9-22—DRNI Gateway and Aggregator machine state diagram**

Following an administrative or protocol change to any parameter affecting the Home_Aggregator_State, Home_Gateway_State, or Home_Gateway_Preference variables, the machine enters the UPDATE_HOME state.

After updating the state variables for the Home DRNI Gateway, or when the DRCP Receive machine indicates that there is new state information for the Neighbor DRNI Gateway, the machine enters the

HOME_GW/AGG_SELECTION state. In this state the Home DRNI Gateway's view of the state of the DRNI is used to select that Aggregator Port and DRNI Gateway Port to be used for forwarding each Port Conversation ID and Gateway Conversation ID, respectively.

After updating the Aggregator and DRNI Gateway Port selections, or when the Neighbor has acknowledged a new state for the Home DRNI Gateway, the UPDATE_DISTRIBUTED_RELAY_GW/AGG_MASKS state is entered. This state updates the Boolean masks used in the DRNI Gateway Relay (9.5.2) to forward frames between the Aggregator, Gateway, and Intra-Relay Ports.

### 9.6.13 DRCP Transmit machine

The DRCP Transmit machine shall implement the function specified in Figure 9-23 with its associated parameters (9.6.4 through 9.6.9). There is one DRCP Transmit machine in a DRNI Gateway.

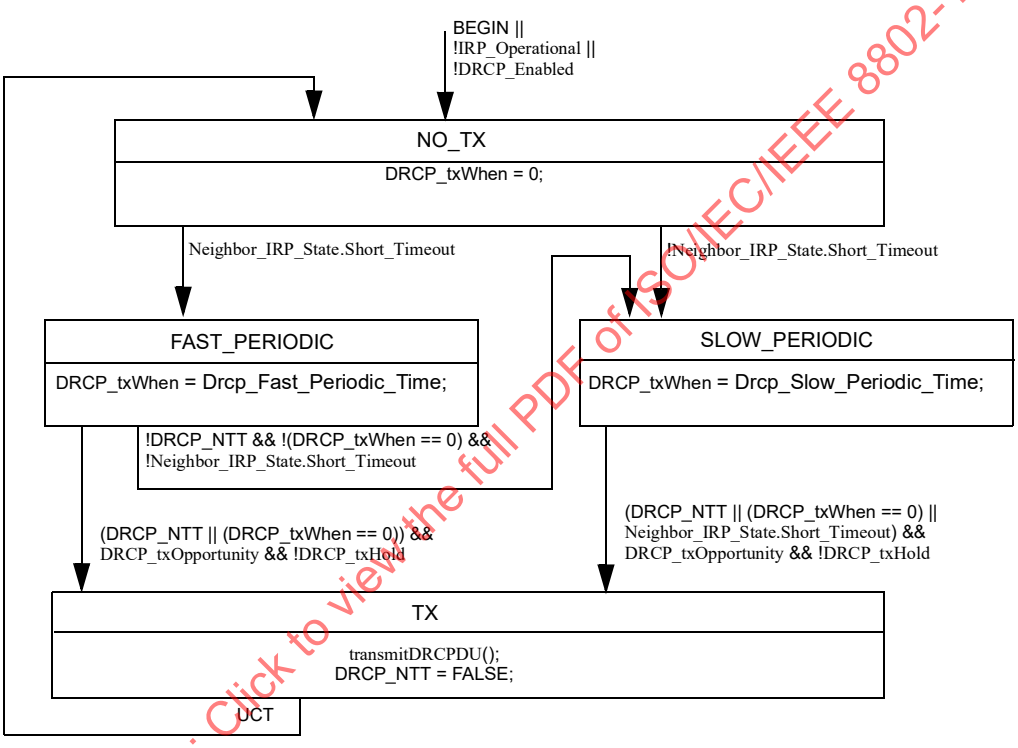



**Figure 9-23—DRCP Transmit state diagram**

The DRCP Transmit state machine has four states:

a) *NO_TX*. While in this state, DRCPDU transmissions are disabled. When IRP_Operational and DRCP_Enabled are TRUE, the state machine transitions to either FAST_PERIODIC or SLOW_PERIODIC depending on the receive timeout rate in use by the Neighbor (indicated by Neighbor_IRP_State.Short_Timeout).

b) *FAST_PERIODIC*. While in this state, periodic DRCPDU transmissions are enabled at a fast transmission rate.

c) *SLOW_PERIODIC*. While in this state, periodic DRCPDU transmissions are enabled at a slow transmission rate.

d) *TX*. This transitory state is entered when DRCP_NTT has been asserted or the periodic transmission timer (DRCP_txWhen) has expired. A single DRCPDU is transmitted. Then the state machine transitions through NO_TX to either FAST_PERIODIC or SLOW_PERIODIC.

# Annex A

(normative)

# Protocol implementation conformance statement (PICS) proforma[10]

## A.1 Introduction

The supplier of an implementation that is claimed to conform to Clause 5 shall complete the following PICS proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. A PICS is included at the end of each clause as appropriate. The PICS can be used for a variety of purposes by various parties, including the following:

a) As a checklist by the protocol implementer, to reduce the risk of failure to conform to the standard through oversight;
b) As a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma, by the supplier and acquirer, or potential acquirer, of the implementation;
c) As a basis for initially checking the possibility of interworking with another implementation by the user, or potential user, of the implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICS);
d) As the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation, by a protocol tester.

### A.1.1 Abbreviations and special symbols

The following symbols are used in the PICS proforma:

| | |
|---|---|
| M | mandatory field/function |
| ! | negation |
| O | optional field/function |
| O.<n> | optional field/function, but at least one of the group of options labeled by the same numeral <n> is required |
| O/<n> | optional field/function, but one and only one of the group of options labeled by the same numeral <n> is required |
| X | prohibited field/function |
| <item>: | simple-predicate condition, dependent on the support marked for <item> |
| <item1>*<item2>: | AND-predicate condition, the requirement shall be met if both optional items are implemented |

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

### A.1.2 Instructions for completing the PICS proforma

The first part of the PICS proforma, Implementation Identification and Protocol Summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire divided into subclauses, each containing a group of items. Answers to the questionnaire items are to be provided in the right-most column, either by simply marking an answer to indicate a restricted choice (usually Yes, No, or Not Applicable), or by entering a value or a set or range of values. (Note that there are some items where two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column; the second column contains the question to be answered; the third column contains the reference or references to the material that specifies the item in the main body of the standard; the sixth column contains values and/or comments pertaining to the question to be answered. The remaining columns record the status of the items—whether the support is mandatory, optional or conditional—and provide the space for the answers.

The supplier can also provide, or be required to provide, further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled A<i> or X<i>, respectively, for cross-referencing purposes, where <i> is any unambiguous identification for the item (e.g., simply a numeral); there are no other restrictions on its format or presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the protocol implementation conformance statement for the implementation in question.

Note that where an implementation is capable of being configured in more than one way, according to the items listed under Major Capabilities/Options, a single PICS might be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's configuration capabilities, if that would make presentation of the information easier and clearer.

### A.1.3 Additional information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and the PICS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations; or a brief rationale, based perhaps upon specific application needs, for the exclusion of features that, although optional, are nonetheless commonly present in implementations.

References to items of Additional Information can be entered next to any answer in the questionnaire, and can be included in items of Exception Information.

### A.1.4 Exceptional information

It occasionally happens that a supplier will wish to answer an item with mandatory or prohibited status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this; instead, the supplier is required to write into the Support column an X<i> reference to an item of Exception Information, and to provide the appropriate rationale in the Exception item itself.

An implementation for which an Exception item is required in this way does not conform to this standard.

Note that a possible reason for the situation described above is that a defect in the standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

### A.1.5 Conditional items

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself, and its status if it does apply—mandatory, optional, or prohibited—are dependent upon whether certain other items are supported.

Individual conditional items are indicated by a conditional symbol of the form "<item>:<s>" in the Status column, where "<item>" is an item reference that appears in the first column of the table for some other item, and "<s>" is a status symbol, M (Mandatory), O (Optional), or X (Not Applicable).

If the item referred to by the conditional symbol is marked as supported, then (1) the conditional item is applicable, (2) its status is given by "<s>", and (3) the support column is to be completed in the usual way; otherwise, the conditional item is not relevant and the Not Applicable (N/A) answer is to be marked.

Each item whose reference is used in a conditional symbol is indicated by an asterisk in the Item column.

### A.1.6 Identification

#### A.1.6.1 Implementation identification

| | |
|---|---|
| Supplier (Note 1) | |
| Contact point for queries about the PICS (Note 1) | |
| Implementation Name(s) and Version(s) (Notes 1 and 3) | |
| Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names (Note 2) | |
| NOTE 1—Required for all implementations.<br>NOTE 2—May be completed as appropriate in meeting the requirements for the identification.<br>NOTE 3—The terms Name and Version are interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model). | |

#### A.1.6.2 Protocol summary

| | |
|---|---|
| Identification of protocol specification | IEEE Std 802.1AX-2020, IEEE Standard for Local and Metropolitan Area Networks—Link Aggregation |
| Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS | |

## A.2 PICS proforma for Clause 6

### A.2.1 Major capabilities/options

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | The items below are relevant to the ports of a System for which support is claimed | | | | |
| LA | Is Link Aggregation supported as specified in Clause 6 ? | Item a) in 5.3, 6.2 | Support the Link Aggregation Sublayer and conform to the state machines and procedures in 6.2 | M | Yes [ ] |
| LACP | Is the Link Aggregation Control Protocol supported? | Item b) in 5.3, 6.3, 6.4 | Support the Link Aggregation Control Protocol and conform to the state machines and procedures in 6.3 and 6.4 | M | Yes [ ] |
| V2LA | Are Version 2 LACPDUs supported? | Item c) in 5.3, 6.4.2 | Transmit and receive LACPDUs with Version Number equal to 2 as specified in 6.4.2 | M | Yes [ ] |
| MG | Is the Marker Generator/ Receiver supported? | Item a) in 5.3.2, 6.2.5 | | O | Yes [ ]<br>No [ ] |
| MGT | Is Management supported? | Item b) in 5.3.2, Clause 7 | Support the management functionality for Link Aggregation as specified in Clause 7 | O | Yes [ ]<br>No [ ] |
| MIB | Does the implementation support management operations using SMIv2 MIB modules? | Item c) in 5.3.2, Annex D | Support SMIv2 MIB modules for the management of Link Aggregation capabilities | MGT: O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| AM | Is there Aggregation Port Debug Information package support? | Item d) in 5.3.2 | | MGT: O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| PSFD | Is Per-Service Frame Distribution supported? | Item i) in 5.3.2, 8.2 | | O | Yes [ ]<br>No [ ] |
| CSCD | Is Conversation-Sensitive Collection and Distribution supported? | Item h) in 5.3.2 | | PSFD: M | N/A [ ]<br>Yes [ ] |
| CCR | Does the Aggregation System have configuration restrictions on the number of Aggregation Ports that can be aggregated? | 6.7.1, 6.7.2 | | O | Yes [ ]<br>No [ ] |
| DRNI | Is DRNI supported? | 5.4 | | O | Yes [ ]<br>No [ ] |

### A.2.2 LLDP Port connectivity

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| LLDP | Does the System support LLDP? | Item e) in 5.3.2, 6.2 | | O | Yes [ ]<br>No [ ] |
| LLDPPM | Does the System support the LLDP Parser/Multiplexer on the Aggregation Ports? | Item e) in 5.3.2, F.1.1 | | LLDP: M | N/A [ ]<br>Yes [ ] |

### A.2.3 Protocol Parser/Multiplexer support

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| PPM | Does the System support a Protocol Parser/Multiplexer for a protocol that is supported by the System but not specified in this standard? | Item f) in 5.3.2, 6.1.3 | | O | N/A []<br>Yes [ ] |

### A.2.4 Frame Collector

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| FC1 | Frame Collector function | 6.2.3 | As specified in the state machine shown in Figure 6-4 and associated definitions in 6.2.3.1 | M | Yes [ ] |
| FC2 | Frame Collector function—CollectorMaxDelay | 6.2.3.1.4 | Deliver or discard frames within CollectorMaxDelay | M | Yes [ ] |

### A.2.5 Frame Distributor

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | Distribution algorithm ensures the following, when frames received by Frame Collector: | 6.2.4 | | | |
| FD1 | Frame misordering | | None | M | Yes [ ] |
| FD2 | Frame duplication | | None | M | Yes [ ] |
| FD3 | Frame Distributor function | 6.2.4 | Function as specified in the state machine shown in Figure 6-5 and associated definitions in 6.2.4.1 | M | Yes [ ] |

### A.2.6 Marker protocol

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| MGR1 | Marker Generator/Receiver | 6.2.5 | | MG:M | N/A [ ]<br>Yes [ ] |
| MGR2 | Marker Responder | Item d) in 5.3, 6.2.6 | Function specified in 6.5.4.2 | M | Yes [ ] |

### A.2.7 Aggregator Parser/Multiplexer

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| APM1 | Aggregator Multiplexer | 6.2.7 | Transparent pass-through of frames | M | Yes [ ] |
| APM2 | Aggregator Multiplexer | 6.2.7 | Function specified by state machine shown in Figure 6-7 and associated definitions in 6.2.7.1 | M | Yes [ ] |
| APM3 | Aggregator Parser | 6.2.7 | Function specified by state machine shown in Figure 6-6 and associated definitions in 6.2.7.1 | M | Yes [ ] |
| APM4 | Aggregator Parser | 6.2.7 | Discard of RX frames when Aggregation Port not Collecting | M | Yes [ ] |

### A.2.8 LACP Parser/Multiplexer

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| CPM1 | LACP Multiplexer | 6.2.9 | Transparent pass-through of frames | M | Yes [ ] |
| CPM2 | LACP Parser | 6.2.9 | Function specified by state machine shown in Figure 6-2 and associated definitions in 6.2.9 | M | Yes [ ] |

### A.2.9 System identification

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| SID1 | Globally unique identifier | 6.3.2 | Globally administered individual MAC address plus System Priority | M | Yes [ ] |
| SID2 | MAC address chosen | 6.3.2 | MAC address associated with one of the Aggregation Ports | O | Yes [ ]<br>No [ ] |

### A.2.10 Aggregator identification

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| AID1 | Globally unique identifier | 6.3.3 | Globally administered individual MAC address | M | Yes [ ] |
| AID2 | Integer identifier | 6.3.3 | Uniquely identifies the Aggregator within the System | M | Yes [ ] |
| *AID3 | Unique identifier allocated | 6.3.3 | Unique identifier assigned to one of its attached Aggregation Ports | O | Yes [ ]<br>No [ ] |
| AID4 | | | Unique identifier not assigned to any other Aggregator | !AID3 :M | N/A [ ]<br>Yes [ ] |

### A.2.11 Port identification

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| PID1 | Port Identifiers | 6.3.4 | Unique within a System; Port Number 0 not used for any Aggregation Port | M | Yes [ ] |

### A.2.12 Capability identification

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| CID1 | Administrative and operational Key values associated with each Aggregation Port | 6.3.5 | | M | Yes [ ] |
| CID2 | Administrative and operational Key values associated with each Aggregator | 6.3.5 | | M | Yes [ ] |

### A.2.13 Link Aggregation Group identification

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| LAG1 | LAG ID component values | 6.3.6.1 | Actor's values nonzero. Partner's admin values only zero for Solitary Aggregation Ports | M | Yes [ ] |

### A.2.14 Detaching a link from an Aggregator

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| DLA1 | Effect on conversation reallocated to a different link | 6.3.14 | Frame ordering preserved | M | Yes [ ] |

### A.2.15 LACPDU structure

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| LPS2 | LACPDU structure | 6.4.2 | As shown in Figure 6-8 and as described in 6.4.2 | M | Yes [ ] |
| LPS3 | LACPDU structure | 6.4.2 | All Reserved octets ignored on receipt and transmitted as zero | M | Yes [ ] |

### A.2.16 Receive machine

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| RM1 | LACP Receive machine | 6.4.11 | As defined in Figure 6-14 and associated parameters | M | Yes [ ] |
| RM2 | Validation of LACPDUs | 6.4.8 | As specified in the recordPDU function | M | Yes [ ] |

### A.2.17 Selection Logic

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | Selection Logic requirements | 6.4.12.1 | | | |
| SLM1 | Aggregator support | | At least one Aggregator per System | M | Yes [ ] |
| SLM2 | Aggregation Port Keys | | Each Aggregation Port assigned an operational Key | M | Yes [ ] |
| SLM3 | Aggregator Keys | | Each Aggregator assigned an operational Key | M | Yes [ ] |
| SLM4 | Aggregator Identifiers | | Each Aggregator assigned an identifier | M | Yes [ ] |
| SLM5 | Aggregator selection | | If same Key assignment as Aggregation Port | M | Yes [ ] |
| SLM6 | Aggregation Ports that are members of the same LAG | | Aggregation Ports select same Aggregator | M | Yes [ ] |
| SLM7 | Pair of Aggregation Ports connected in loopback | | Not select same Aggregator as each other | M | Yes [ ] |
| SLM8 | Aggregation Port is not Aggregateable | | Not select same Aggregator as any other Aggregation Port | M | Yes [ ] |
| SLM9 | Aggregation Port is Aggregateable | | Not select same Aggregator as any non-Aggregateable Aggregation Port | M | Yes [ ] |
| SLM10 | Aggregation Port unable to select an Aggregator | | Aggregation Port not attached to any Aggregator | M | Yes [ ] |
| SLM11 | Further aggregation constraints | | Aggregation Ports can be selected as standby | O | Yes [ ]<br>No [ ] |
| SLM12 | Selected variable | | Set to SELECTED or STANDBY once Aggregator is determined | M | Yes [ ] |
| SLM13 | Recommended default operation of Selection Logic | 6.4.12.2 | Meets requirements of 6.4.12.2 | O | Yes [ ]<br>No [ ] |

### A.2.18 Mux machine

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| XM1 | Mux machine | 6.4.13 | As defined in Figure 6-16 and associated parameters | M | Yes [ ] |

### A.2.19 Transmit machine

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| TM1 | LACP Transmit machine | 6.4.14 | As defined in Figure 6-17 and associated parameters | M | Yes [ ] |
| TM2 | Include Port Algorithm TLV | 6.4.2.4.1 | | CSCD:M | N/A [ ]<br>Yes [ ] |
| TM3 | Include Port Conversation Link Digest TLV | 6.4.2.4.2 | When Actor_Oper_Port_Algorithm is not "Unspecified" (Table 8-1) | CSCD:M | N/A [ ]<br>Yes [ ] |
| TM4 | Include Port Conversation Service Digest TLV | 6.4.2.4.2 | When MSB of last octet of Actor_Oper_Port_Algorithm is 1 | CSCD:M | N/A [ ]<br>Yes [ ] |

### A.2.20 Marker protocol

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| FP1 | Respond to all received Marker PDUs | Item d) in 5.3, 6.5.1 | As specified by 6.5.4 | M | Yes [ ] |
| FP2 | Use of the Marker protocol | 6.5.1 | As specified by 6.5.4 | O | Yes [ ]<br>No [ ] |
| FP3 | MARKER.request service primitives request rate | 6.5.4.1 | Maximum of five during any one-second period | MG:M | N/A [ ]<br>Yes [ ] |
| FP6 | Marker PDU structure | 6.5.3.3 | As shown in Figure 6-19 and as described | MG:M | N/A [ ]<br>Yes [ ] |
| FP7 | Marker Response PDU structure | Item d) in 5.3, 6.5.3.3 | As shown in Figure 6-19 and as described | M | Yes [ ] |
| FP8 | Marker Responder state machine | Item d) in 5.3, 6.5.4.2 | As specified in Figure 6-20 and 6.5.4.2.1–6.5.4.2.2 | M | Yes [ ] |
| FP9 | Validation of Marker Request PDUs | Item d) in 5.3, 6.5.4.2.2 | Marker Responder shall not validate the Version Number, Pad, or Reserved fields | M | Yes [ ] |

### A.2.21 Management

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | If MGT is not supported, mark N/A and ignore the remainder of this table | | | | N/A [ ] |
| MGT1 | Is the Basic package supported? | Table 7–1, 7.3.1.1.1, 7.3.2.1.1 | Support of the Managed objects marked as members of the Basic package in Table 7–1 | MGT: M | N/A [ ]<br>Yes [ ] |
| MGT2 | Is the Mandatory package supported? | Table 7–1, 7.3.1.1.2–7.3.1.1.16, 7.3.1.1.19, 7.3.1.1.20, 7.3.1.1.31, 7.3.1.1.32, 7.3.1.2.1, 7.3.1.2.2, 7.3.2.1.2–7.3.2.1.24, 7.3.2.1.30–7.3.2.1.34, 7.3.2.2.1 | Support of the Managed objects marked as members of the Mandatory package in Table 7–1 | MGT: M | N/A [ ]<br>Yes [ ] |
| MGT3 | Is the Recommended package supported? | Table 7–1, 7.3.1.1.17, 7.3.1.1.18, 7.3.1.1.25–7.3.1.1.30 | Support of the Managed objects marked as members of the Recommended package in Table 7–1 | MGT: O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| MGT4 | Is the Optional package supported? | Table 7–1, 7.3.1.1.21–7.3.1.1.24 | Support of the Managed objects marked as members of the Optional package in Table 7–1 | MGT: O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| MGT5 | Is the Aggregation Port Statistics package supported? | Table 7–1, 7.3.3.1.1–7.3.3.1.9 | Support of the Managed objects marked as members of the Aggregation Port Statistics package in Table 7–1 | MGT: O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| MGT6 | Is the Aggregation Port Debug Information package supported? | Table 7–1, 7.3.4.1.1–7.3.4.1.9 | Support of the Managed objects marked as members of the Aggregation Port Debug Information package in Table 7–1 | MGT: O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| MGT7 | Is the Per-Service Frame Distribution package supported? | Table 7–1, 7.3.1.1.33–7.3.1.1.43, 7.3.2.1.25–7.3.2.1.29 | Support of the Managed objects marked as members of the Per-Service Frame Distribution package in Table 7–1 | MGT AND PSFD: M | N/A [ ]<br>Yes [ ] |
| MGT8 | Is the DRNI package supported? | Table 7–1, 7.4.1.1.1–7.4.1.1.47 | Support of the Managed objects marked as members of the DRNI package in Table 7–1 | MGT AND DRNI: M | N/A [ ]<br>Yes [ ] |

### A.2.22 Per-Service Frame Distribution

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | If PSFD is not supported, mark N/A and ignore the remainder of this table | | | | N/A [ ] |
| PSFD1 | Is Frame Distribution by VID supported? | 8.2.1, 8.2.2 | Distribution based on C-VLAN Identifier or S-VLAN Identifier | PSFD: M | N/A [ ]<br>Yes [ ] |
| PSFD2 | Is Frame Distribution by I-SID supported? | 8.2.3 | | PSFD: O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| PSFD3 | Does the implementation support Frame Distribution by other methods not specified by this standard? | | | PSFD: O | N/A [ ]<br>Yes [ ]<br>No [ ] |

### A.2.23 Conversation-sensitive frame collection and distribution

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | If CSCD is not supported, mark N/A and ignore the remainder of this table | | | | N/A [ ] |
| CSCD1 | Are the Conversation-sensitive LACP variables and functions supported? | 6.6.3 | | CSCD: M | N/A [ ]<br>Yes [ ] |
| CSCD2 | Is the Update Mask state diagram supported? | 6.6.3 | As shown in Figure 6-23 and as described | CSCD: M | N/A [ ]<br>Yes [ ] |
| CSCD3 | Are the values of FORCE_TRUE and AUTO supported for Admin_Discard_Wrong_Conversation? | 6.6.3 | Item h) in 5.3.2, 6.6.3.1, 7.3.1.1.35 | CSCD: O | N/A [ ]<br>Yes [ ]<br>No [ ] |

### A.2.24 Configuration capabilities and restrictions

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| CCR1 | Algorithm used to determine subset of Aggregation Ports that can be aggregated in Systems that have limited aggregation capability | 6.7.1 | As specified in items a) to e) of 6.7.1 | CCR:M | N/A [ ]<br>Yes[ ] |
| CCR2 | Key value modification to generate optimum aggregation | 6.7.2 | | CCR:O | N/A [ ]<br>Yes [ ]<br>No [ ] |
| CCR3 | Key value modification when System has higher System Aggregation Priority | | | CCR2:M | N/A [ ]<br>Yes [ ] |
| CCR4 | Key value modification when System has lower System Aggregation Priority | | | CCR2:X | N/A [ ]<br>No [ ] |
| CCR5 | Support the LACP configuration for dual-homed Systems | 6.7.5 | | M | Yes[ ] |

### A.2.25 Link Aggregation on shared-medium links

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| LSM1 | Shared-medium links—Configuration | 6.7.3, 6.4.11 | Forced to be Solitary links | M | Yes [ ] |
| LSM2 | Shared-medium links—Operation of LACP | 6.7.3 | LACP is disabled | M | Yes [ ] |

### A.2.26 Distributed Resilient Network Interconnect

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | If DRNI is not supported, mark N/A and ignore the remainder of this table | | | | N/A [ ] |
| DRNI1 | Is the DRNI Gateway and the formation of a DRNI in cooperation with a single other DRNI System, which also conforms to the provisions of this standard for DRNI, supported? | Item c)1) in 5.4, 9.2, 9.5 | Support the state machines and procedures in 9.5 and 9.6 | DRNI:M | Yes [ ] |
| DRNI2 | Can the IRC supported by a dedicated link? | Item d) in 5.4, 9.3 | | DRNI:M | Yes [ ] |
| DRNI3 | Can the IRC be supported by a LAG ? | Item b) in 5.4.2, 9.3 | Have an Aggregator dedicated to supporting the IRC | DRNI:O | Yes [ ]<br>No [ ] |
| DRNI4 | Can the IRC be supported by a method other than a dedicated link or a dedicated Aggregator? | Item c) in 5.4.2, 9.3 | | DRNI:O | Yes [ ]<br>No [ ] |

### A.2.27 DRCPDU structure

| Item | Feature | Subclause | Value/Comment | Status | Support |
|---|---|---|---|---|---|
| | If DRNI1 is not supported, mark N/A and ignore the remainder of this table | | | | N/A [ ] |
| DRST1 | DRCPDU addressing and protocol identifications | 9.6.1 | As described | DRNI1:M | Yes [ ] |
| DRST2 | DRCPDU structure | 9.6.2 | As shown in Figure 9-12 and as described | DRNI1:M | Yes [ ] |
| DRST3 | DRCPDU structure | 9.6.2 | All Reserved octets ignored on receipt and transmitted as zero | DRNI1:M | Yes [ ] |
| DRST4 | Is the Organization-Specific TLV supported? | Item d) in 5.4.2, 9.6.2.7 | As shown in Figure 9-18 and as described | DRNI1:O | Yes [ ]<br>No [ ] |

# Annex B

(informative)

# Collection and distribution algorithms

## B.1 Introduction

The specification of the Frame Collection and Frame Distribution functions was defined with the following considerations in mind:

a) Frame duplication is not permitted.

b) Frame ordering has to be preserved in aggregated links. Strictly, the Internal Sublayer Service specification (IEEE Std 802.1AC) states that order has to be preserved for frames with a given SA, DA, and priority; however, this is a tighter constraint than is absolutely necessary. There can be multiple, logically independent conversations in progress between a given SA-DA pair at a given priority; the real requirement is to maintain ordering within a conversation, though not necessarily between conversations.

c) A single algorithm can be defined for the Frame Collection function that is independent of the distribution algorithm(s) employed by the Partner System.

d) In the interests of simplicity and scalability, the Frame Collection function does not perform reassembly functions, reorder received frames, or modify received frames. Frame Distribution functions, therefore, do not make use of segmentation techniques, do not label or otherwise modify transmitted frames in any way, and have to operate in a manner that will inherently ensure proper ordering of received frames with the specified Frame Collector.

e) The distribution and collection algorithms need to be capable of handling dynamic changes in aggregation membership.

f) There are expected to be many different topologies and many different types of devices in which Link Aggregation can be employed. It is therefore unlikely that a single distribution algorithm would be applicable in all cases.

A simple Frame Collection function has been specified. The Frame Collector preserves the order of frames received on a given link, but does not preserve frame ordering among links. The Frame Distribution function maintains frame ordering by

g) Transmitting frames of a given conversation on a single link at any time.

h) Before changing the link on which frames of a given conversation are transmitted, ensuring that all previously transmitted frames of that conversation have been received to a point such that any subsequently transmitted frames received on a different links will be delivered to the Aggregator Client at a later time.

Given the wide variety of potential distribution algorithms, the normative text in Clause 6 specifies only the requirements that such algorithms have to meet, and not the details of the algorithms themselves. To clarify the intent, this informative annex gives examples of distribution algorithms, when they might be used, and the role of the Marker protocol (6.5) in their operation. The examples are not intended to be either exhaustive or prescriptive; implementers can make use of any distribution algorithms as long as the requirements of Clause 6 are met.

## B.2 Port selection

A distribution algorithm selects the Aggregation Port used to transmit a given frame, such that the same Aggregation Port will be chosen for subsequent frames that form part of the same conversation. The algorithm can make use of information carried in the frame in order to make its decision, in combination with other information associated with the frame, such as its reception Aggregation Port in the case of a Bridge.

The algorithm can assign one or more conversations to the same Aggregation Port; however, it has to not allocate some of the frames of a given conversation to one Aggregation Port and the remainder to different Aggregation Ports. The information used to assign conversations to Aggregation Ports could include (but is not limited to) the following:

a) Source MAC address
b) Destination MAC address
c) Reception Aggregation Port
d) Type of destination address (individual or group MAC address)
e) Ethernet Length/Type value (i.e., protocol identification)
f) VLAN Identifier
g) Higher layer protocol information (e.g., addressing and protocol identification information from the LLC sublayer or above)
h) Combinations of the above

One simple approach applies a hash function to the selected information to generate a Port Number. This produces a deterministic (i.e., history independent) Aggregation Port selection across a given number of Aggregation Ports in an aggregation. However, as it is difficult to select a hash function that will generate a uniform distribution of load across the set of Aggregation Ports for all traffic models, it might be appropriate to weight the Aggregation Port selection in favor of Aggregation Ports that are carrying lower traffic levels. In more sophisticated approaches, load balancing is dynamic; i.e., the Aggregation Port selected for a given set of conversations changes over time, independently of any changes that take place in the membership of the aggregation.

## B.3 Dynamic reallocation of conversations to different Aggregation Ports

It can be necessary for a given conversation or set of conversations to be moved from one Aggregation Port to one or more others, as a result of

a) A change to the Actor_Oper_Port_State.Collecting variable of an Aggregation Port on the LAG; or
b) A decision on the part of the Frame Distributor to redistribute the traffic across the set of Aggregation Ports.

Before moving conversation(s) to a new Aggregation Port, it is necessary to ensure that all frames already transmitted that are part of those conversations have been successfully received. The following procedure shows how the Marker protocol (6.5) can be used to ensure that no misordering of frames occurs:

c) Stop transmitting frames for the set of conversations affected. If the Aggregator Client requests transmission of further frames that are part of this set of conversations, these frames are discarded.
d) Start a timer, choosing the timeout period such that, if the timer expires, the destination System can be assumed either to have received or discarded all frames transmitted prior to starting the timer.

e) Use the Marker protocol to send a Marker PDU on the Aggregation Port previously used for this set of conversations.

f) Wait until either the corresponding Marker Response PDU is received or the timer expires.

g) Restart frame transmission for the set of conversations on the newly selected Aggregation Port.

The appropriate timeout value depends on the connected devices. For example, the recommended maximum Bridge Transit Delay is 1 s; if the receiving device is a Bridge, it can be expected to have forwarded or discarded all frames received more than 1 s ago. The appropriate timeout value for other circumstances could be smaller or larger than this by several orders of magnitude. For example, if the two Systems concerned are high-performance end stations connected via Gigabit Ethernet links, then timeout periods measured in milliseconds might be more appropriate. In order to allow an appropriate timeout value to be determined, the Frame Collector parameter CollectorMaxDelay (see 6.2.3) defines the maximum delay that the Frame Collector can introduce between receiving a frame from an Aggregation Port and either delivering it to the Aggregator Client or discarding it. This value will be dependent upon the particular implementation choices that have been made in a System. As far as the operation of the Frame Collector state machine is concerned, CollectorMaxDelay is a constant; however, a management attribute, aAggCollectorMaxDelay (7.3.1.1.32), is provided that allows interrogation and administrative control of its value. Hence, if a System knows the value of CollectorMaxDelay that is in use by a Partner System, it can set the value of timeout used when flushing a link to be equal to that value of CollectorMaxDelay, plus sufficient additional time to allow for the propagation delay experienced by frames between the two Systems. A value of zero for the CollectorMaxDelay parameter indicates that the delay imposed by the Frame Collector is less than the resolution of the parameter (10 μs). In this case, the delay that has to be considered is the physical propagation delay of the channel. Allowing management manipulation of CollectorMaxDelay permits fine-tuning of the value used in those cases where it can be difficult for the equipment to preconfigure a piece of equipment with a realistic value for the physical propagation delay of the channel.

The Marker protocol provides an optimization that can result in faster reallocation of conversations than would otherwise be possible—without the use of markers, the full timeout period would always have to be used in order to be sure that no frames remained in transit between the local Frame Distributor and the remote Frame Collector. The timeout described recovers from loss of Marker or Marker Response PDUs that can occur.

## B.4 Topology considerations in the choice of distribution algorithm

Figure B-1 gives some examples of different aggregated link scenarios. In some cases, it is possible to use distribution algorithms that use MAC frame information to allocate conversations to links; in others, it is necessary to make use of higher-layer information.

In example A, there is a many-to-many relationship between end stations communicating over the aggregated link. It would be possible for each Bridge to allocate conversations to links simply on the basis of source or destination MAC addresses.

In examples B and C, a number of end stations communicate with a single server via the aggregated link. In these cases, the distribution algorithm employed in the server or in Bridge 2 can allocate traffic from the server on the basis of destination MAC address; however, as one end of all conversations constitutes a single server with a single MAC address, traffic from the end stations to the server would have to be allocated on the basis of source MAC address. These examples illustrate the fact that different distribution algorithms can be used in different devices, as appropriate to the circumstances. The collection algorithm is independent of the distribution algorithm(s) that are employed.

In examples D and E, assuming that the servers are using a single MAC address for all of their traffic, the only appropriate option is for the distribution algorithm used in the servers and Bridges to make use of

higher-layer information (e.g., Transport Layer socket identifiers) in order to allocate conversations to links. Alternatively, in example E, if the servers were able to make use of multiple MAC addresses and allocate conversations to them, then the Bridges could revert to MAC Address-based allocation.

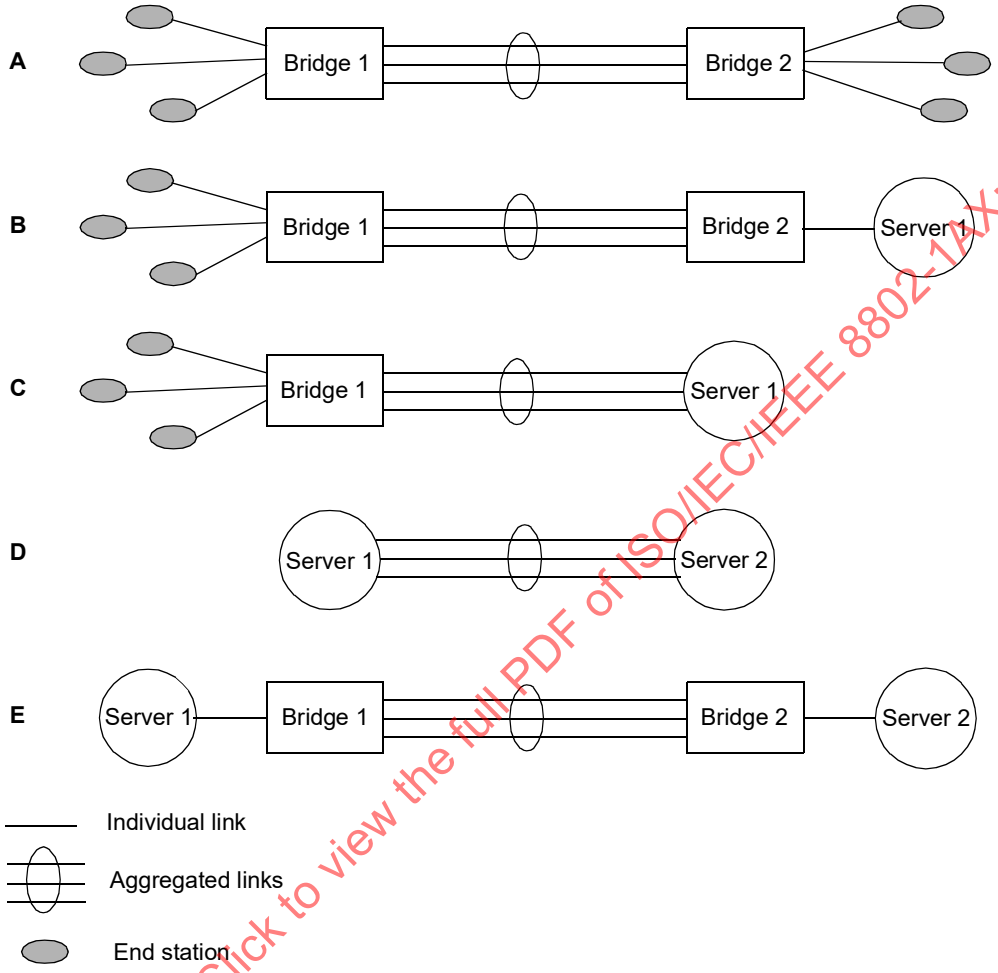



**Figure B-1—Link aggregation topology examples**

# Annex C

(informative)

# LACP standby link selection and dynamic Key management

## C.1 Introduction

While any two Aggregation Ports on a given System that have been assigned the same administrative Key might be capable of aggregation, it is not necessarily the case that an arbitrary selection of such Aggregation Ports can be aggregated. (Keys might have been deliberately assigned to allow one link to be operated specifically as a hot standby for another.) A System might reasonably limit the number of Aggregation Ports attached to a single Aggregator, or the particular way more than two Aggregation Ports can be combined.

In cases where both communicating Systems have constraints on aggregation, it is necessary for them both to agree to some extent on the links to be selected for aggregation and on which not to use; otherwise, it might be possible for the two Systems to make different selections, possibly resulting in no communication at all.

When one or more links have to be selected as standby, it is possible that they could be used as part of a different Link Aggregation Group (LAG). For this to happen, one or another of the communicating Systems has to change the operational Key values used for the Aggregation Ports attached to those links.

If the operational Key values were to be changed independently by each System, the resulting set of aggregations could be unpredictable. It is possible that numerous aggregations, each containing a single link, can result. Worse, with no constraint on changes, the process of both Systems independently searching for the best combination of operational Key values might never end.

This annex describes protocol rules for standby link selection and dynamic Key management. It provides examples of a dynamic Key management algorithm applied to connections between Systems with various aggregation constraints.

## C.2 Goals

The protocol rules presented

a) Enable coordinated, predictable, and reproducible standby link selections.
b) Permit predictable and reproducible partitioning of links into aggregations by dynamic Key management.

They do not require

c) An LACP System to understand all the constraints on aggregations of multiple Aggregation Ports that might be imposed by other Systems.
d) Correct configuration of parameters, i.e., they retain the plug and play attributes of LACP.

## C.3 Standby link selection

Every link between Systems operating LACP is assigned a unique priority. This priority comprises (in priority order) the System Priority, System MAC Address, Port Priority, and Port Number of the higher-priority System. In priority comparisons, numerically lower values have higher priority.

Aggregation Ports are considered for active use in an aggregation in link priority order, starting with the Aggregation Port attached to the highest priority link. Each Aggregation Port is selected for active use if preceding higher priority selections can also be maintained; otherwise, the Aggregation Port is selected as standby.

## C.4 Dynamic Key management

Dynamic Key management changes the Key values used for links that either System has selected as a standby to allow use of more links. Whether this is desirable depends on their use. For example, if a single spanning tree is being used throughout the network, separating standby links into a separate aggregation serves little purpose. In contrast, if equal cost load sharing is being provided by routing, making additional bandwidth available in a separate LAG might be preferable to holding links in standby to provide link resilience.

The communicating System with the higher priority (as determined by the System Identifier) controls dynamic Key changes. Dynamic Key changes can be made only by this controlling System.

NOTE—The controlling System can observe the Port Priorities assigned by the Partner System, if it wishes to take these into account.

This rule prevents the confusion that could arise if both Systems change Keys simultaneously. In principle the controlling System might search all possible Key combinations for the best way to partition the links into groups. In practice the number of times that Keys have to be changed to yield acceptable results is small.

After each Key change, the controlling System assesses which links are being held in standby by its Partner. Although there is no direct indication of this decision, the synchronization state of standby links will be FALSE. After matched information is received from the protocol Partner, and before acting on this information, a “settling time” allows for the Partner’s aggregate wait delay, and for the selected links to be aggregated. Twice the Aggregate Wait Time (the expiry period for the waitWhile timer), i.e., 4 s, is sufficient. If matched Partner information indicates that all the links that the Actor can make active have been synchronized, it can proceed to change Keys on other links without further delay.

## C.5 A dynamic Key management algorithm

The following algorithm is simple but effective.

After the “settling time” (see C.4) has elapsed, the controlling System scans its Aggregation Ports in the LAG (i.e., all those Aggregation Ports with a specific operational Key value that have the same Partner System Priority, System MAC Address, and Key) in descending priority order.

For each Aggregation Port, it might wish to know

a) Is the Aggregation Port (i.e., the Actor) *capable* of being aggregated with the Aggregation Ports already selected for aggregation with the current Key? Alternatively, is the Actor *not capable* of this aggregation?
b) Is the Aggregation Port’s synchronization state TRUE or FALSE?

As it inspects each Aggregation Port, it can

c) *Select* the Aggregation Port to be part of the aggregation with the current Key.
d) *Retain* the current Key for a further iteration of the algorithm, without selecting the Aggregation Port to be part of the current aggregation.
e) *Change* the operational Key to a new value. Once a new value is chosen, all the Aggregation Ports in the current LAG that have their Keys changed will be changed to this new value.

As the Aggregation Ports are scanned for the first time,

1) The highest priority Aggregation Port is always selected.

   If it is capable and Actor_Oper_Port_State.Synchronization is TRUE, move to step 2).

   Otherwise, **change** the operational Key of all other Aggregation Ports (if any) in this LAG, and apply this dynamic Key algorithm to those Aggregation Ports, beginning with step 1), after the settling time.

2) Move to the next Aggregation Port.

   If there is a next Aggregation Port, continue at step 3).

   Otherwise, dynamic Key changes for Aggregation Ports with this operational Key are complete.

   Note that Aggregation Ports that were once in the same aggregation might have had their operational Keys changed to (further) new values. If so, apply the dynamic Key management algorithms to those Aggregation Ports, beginning with step 1), after the settling time.

3) If this Aggregation Port is capable and Actor_Oper_Port_State.Synchronization is TRUE, then select it, and repeat from step 2).

   If this Aggregation Port has Actor_Oper_Port_State.Synchronization FALSE, then change the operational Key, and repeat from step 2).

   If this Aggregation Port is not capable but Actor_Oper_Port_State.Synchronization is TRUE, then **change** the operational Key, and move to step 4).

4) Move to the next Aggregation Port.

   If there is a next Aggregation Port, continue at step 5).

   Otherwise, if there are still Aggregation Ports in the current LAG (which will have the current operational Key), then wait for the settling time, and apply the dynamic Key management algorithm, beginning with the first such Aggregation Port, at step 3).

   Otherwise, dynamic Key changes for Aggregation Ports with this operational Key are complete.

5) **If** this Aggregation Port is capable, then retain the current Key, and repeat from step 2).
   Otherwise, **change** the operational Key, and repeat from step 2).

This procedure is repeated until no links with Actor_Oper_Port_State.Synchronization FALSE remain, or a limit on the number of steps has been reached.

If the Partner's System Identifier changes on any link at any time, then the Actor's operational Key for that link reverts to the administrative Key value, and the dynamic Key procedure is rerun. This can involve changing the operational Key values for all the links that were assigned Key values subsequent to the change in Key for the link with the new Partner.

## C.6 Example 1

Two Systems, A and B, are connected by four parallel links. Each System can support a maximum of two links in an aggregation. They are connected as shown in Figure C-1. System A is the higher priority System.

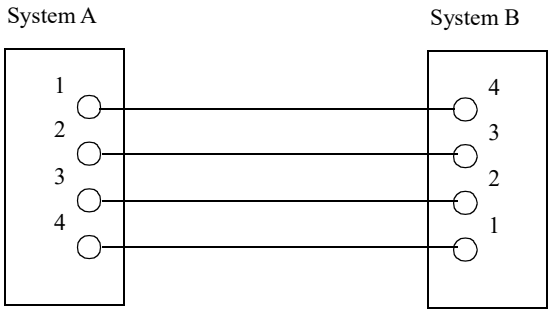



**Figure C-1—Example 1**

The administrative Key for all of System A and System B's Aggregation Ports is 1. Neither System knows before the configuration is chosen that all its Aggregation Ports would attach to links of the same Partner System. Equally, if the links were attached to two different Systems, it is not known which pair of links (e.g., 1 and 2, or 1 and 4) would be attached to the same Partner. So choosing the administrative Keys values to be identical for four Aggregation Ports, even though only two could be actively aggregated, is very reasonable.

If there was no rule for selecting standby links, then System A and System B might have both selected their own Aggregation Ports 1 and 2 as the active links, and there would be no communication. With the rule, the links A1-B4 and A2-B3 will become active, while A3-B2 and A4-B1 will be standby.

Since System A is the higher-priority System, System B's operational Key values will remain 1 while System A might dynamically change Keys, though it can choose to retain the standby links. Following the Key management algorithm suggested, System A would be able to change the Keys for A3 and A4 in a little over 2 s (depending on how fast System B completes the process of attaching its Aggregation Ports to the selected Aggregator) after the connections were first made, and both aggregations could be operating within 5 s.

If System A's aggregations were to be constrained to a maximum of three links, rather than two, while System B's are still constrained to two, the suggested algorithm would delay for 4 s before changing Keys. Both aggregations could be operating within 7 s.

## C.7 Example 2

A System has the odd design constraint that each of its four Aggregation Ports can be aggregated with one other as follows:

a) Aggregation Port 1 with Aggregation Port 2, or Aggregation Port 4.
b) Aggregation Port 2 with Aggregation Port 3, or Aggregation Port 1.
c) Aggregation Port 3 with Aggregation Port 4, or Aggregation Port 2.
d) Aggregation Port 4 with Aggregation Port 1, or Aggregation Port 3.

This is equivalent to each Aggregation Port being able to aggregate with either Neighbor, understanding the Aggregation Ports to be arranged in a circle.

Two such Systems are connected with four parallel links as shown in Figure C-2.

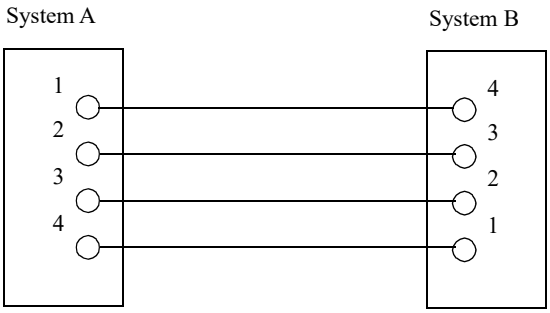



**Figure C-2—Example 2a**

Just as for Example 1, links A1-B4 and A2-B3 become active without changing the operational Key from its original administrative value. The Key for A3 and A4 is changed as soon as they become active, and a few seconds later A3-B2 and A4-B1 become active in a separate aggregation.

If the two Systems had been connected as shown in Figure C-3, the following would occur, assuming that System A operates the simple algorithm already described:

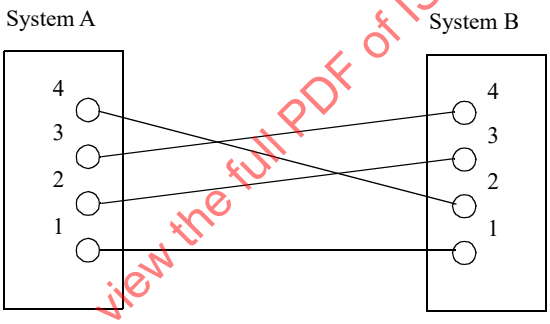



**Figure C-3—Example 2b**

Initially System A advertises an operational Key equal to the administrative Key value of 1 on all Aggregation Ports. System B first selects B1 as active; since the link connects to A1 it has the highest priority. The next highest priority link is B3-A2, but System B cannot aggregate B3 with B1, so System B makes this Aggregation Port standby. System B can aggregate B4-A3, so the Aggregation Port is made active. Finally if B4 is aggregated with B1, B2 cannot be aggregated, so B2 is made standby.

System A, observing the resulting synchronization status from System B, assigns a Key value of 2 to Aggregation Ports 2 and 3, retaining the initial Key of 1 for Aggregation Ports 1 and 4. System B will remove B4 from the aggregation with B1, and substitute B2. B3 and B4 will be aggregated. In the final configuration A1-B1 and A4-B2 are aggregated, as are A2-B3 and A3-B4.

# Annex D

(normative)

# SMIv2 MIB definitions for Link Aggregation[11]

## D.1 Introduction

This annex defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP based internets. In particular it defines objects for managing the operation of the Link Aggregation sublayer, based on the specification of Link Aggregation contained in this standard. This annex includes an MIB module that is SNMPv2 SMI compliant.

## D.2 SNMP Management Framework

For a detailed overview of the documents that describe the current Internet Standard Management Framework, please refer to Section 7 of IETF RFC 3410 (Dec. 2002).

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB modules are defined using the mechanisms defined in the Structure of Management Information (SMI). This clause specifies a MIB module that is compliant to the SMIv2, which is described in IETF STD 58, comprising IETF RFC 2578, IETF RFC 2579, and IETF RFC 2580.

## D.3 Security considerations

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects can be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

This MIB module relates to systems that provide resilient transport at very high capacities based on link aggregation. Improper manipulation of the following tables and the objects in the tables can cause network instability and result in loss of service for a large number of end users.

dot3adAggTable
dot3adAggXTable
dot3adAggPortTable
dot3adAggPortXTable
dot3adAggPortSecondXTable

In particular, the object dot3adAggPortProtocolDA of the dot3adAggPortXTable sets the DA for LACP frames and could be manipulated by an attacker to force a misconfiguration error causing the LAG to fail.

[11] *Copyright release for SMIv2 MIB:* Users of this standard may freely reproduce the SMIv2 MIB in this annex so it can be used for its intended purpose.

Manipulation of the following objects can cause frames to be directed to the wrong ports, which can cause loss of connectivity due to excessive congestion losses, as well as spoiling some or all of the goals of Clause 8, such as bidirectional congruity.

dot3adAggConversationAdminLinkTable
dot3adAggAdminServiceConversationMapTable

For a system implementing Distributed Resilient Network Interconnects (DRNIs), improper manipulation of the following tables and the objects in the tables can have similar negative effects.

dot3adDrniTable
dot3adDrniAdminGatewayConvServiceMapTable

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) can be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables whose objects contain information that can be used to gain sensitive information, which can be used to damage the organization providing the link aggregation services or the users of those services. Sensitive information could be used by an attacker to determine which attacks might be useful to attempt against a given device or to detect whether attacks are being blocked or filtered.

dot3adAggPortStatsTable
dot3adAggPortDebugTable
dot3adAggPortDebugXTable

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control for who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

Implementations should provide the security features described by the SNMPv3 framework (see IETF RFC 3410), and implementations claiming compliance to the SNMPv3 standard shall include full support for authentication and privacy via the User-based Security Model (USM) (see IETF RFC 3414) with the AES cipher algorithm (see IETF RFC 3826 [B4]). Implementations may also provide support for the Transport Security Model (TSM) (see IETF RFC 5591 [B5]) and the View-based Access Control Model (see IETF RFC 3415) in combination with a secure transport such as SSH (see IETF RFC 5592 [B6]) or TLS/DTLS (see IETF RFC 6353 [B8]).

Further, implementers should not deploy SNMP versions prior to SNMPv3. Instead, implementers should deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.



## D.4 Structure of the MIB module

A single MIB module is defined in this annex. Objects in the MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. The overall structure and assignment of objects to their subtrees and their corresponding definitions in Clause 7 is shown in the following subclauses.

### D.4.1 Relationship to the managed objects defined in Clause 7

This subclause contains cross references to the objects defined in Clause 7. Table D.1 contains cross references for the MIB module objects defined in this annex. Table D.2 contains definitions of ifTable elements, as defined in IETF RFC 2863, for an Aggregator. Table D.3 contains definitions of ifTable elements, as defined in IETF RFC 2863, for an DRNI Gateway. These table elements are cross referenced to the corresponding definitions in Clause 7.

**Table D.1—Managed object cross reference table**

| Definition in Clause 7 | MIB object |
|---|---|
| 7.3.1.1.1 aAggID | dot3adAggIndex (ifIndex value) |
| 7.3.1.1.4 aAggActorSystemID | dot3adAggActorSystemID |
| 7.3.1.1.5 aAggActorSystemPriority | dot3adAggActorSystemPriority |
| 7.3.1.1.6 aAggAggregateOrIndividual | dot3adAggAggregateOrIndividual |
| 7.3.1.1.7 aAggActorAdminKey | dot3adAggActorAdminKey |
| 7.3.1.1.8 aAggActorOperKey | dot3adAggActorOperKey |
| 7.3.1.1.10 aAggPartnerSystemID | dot3adAggPartnerSystemID |
| 7.3.1.1.11 aAggPartnerSystemPriority | dot3adAggPartnerSystemPriority |
| 7.3.1.1.12 aAggPartnerOperKey | dot3adAggPartnerOperKey |
| 7.3.1.1.30 aAggPortList | dot3adAggPortListTable |
| 7.3.1.1.32 aAggCollectorMaxDelay | dot3adAggCollectorMaxDelay |
| 7.3.1.1.33 aAggPortAlgorithm | dot3adAggPortAlgorithm |
| -- deprecated -- | dot3adAggPartnerAdminPortAlgorithm |
| 7.3.1.1.34 aAggAdminConvLinkMap | dot3adAggConversationAdminLinkTable |
| -- deprecated -- | dot3adAggPartnerAdminPortConversationListDigest |
| 7.3.1.1.35 aAggAdminDiscardWrongConversation | dot3adAggAdminDiscardWrongConversation2 |
| 7.3.1.1.36 aAggAdminConvServiceMap | dot3adAggAdminServiceConversationMapTable |
| -- deprecated -- | dot3adAggPartnerAdminConvServiceMappingDigest |
| 7.3.1.1.37 aAggOperDiscardWrongConversation | dot3adAggOperDiscardWrongConversation |
| 7.3.1.1.38 aAggConvLinkDigest | dot3adAggConvLinkDigest |
| 7.3.1.1.39 aAggConvServiceDigest | dot3adAggConvServiceDigest |
| 7.3.1.1.40 aAggPartnerPortAlgorithm | dot3adAggPartnerPortAlgorithm |
| 7.3.1.1.41 aAggPartnerConvLinkDigest | dot3adAggPartnerConvLinkDigest |
| 7.3.1.1.42 aAggPartnerConvServiceDigest | dot3adAggPartnerConvServiceDigest |
| 7.3.1.1.43 aAggActiveLinks | dot3adAggActiveLinks |
| 7.3.2.1.1 aAggPortID | dot3adAggPortIndex (ifIndex value) |
| 7.3.2.1.2 aAggPortActorSystemPriority | dot3adAggPortActorSystemPriority |

**Table D.1—Managed object cross reference table *(continued)***

| Definition in Clause 7 | MIB object |
|---|---|
| 7.3.2.1.3 aAggPortActorSystemID | dot3adAggPortActorSystemID |
| 7.3.2.1.4 aAggPortActorAdminKey | dot3adAggPortActorAdminKey |
| 7.3.2.1.5 aAggPortActorOperKey | dot3adAggPortActorOperKey |
| 7.3.2.1.6 aAggPortPartnerAdminSystemPriority | dot3adAggPortPartnerAdminSystemPriority |
| 7.3.2.1.7 aAggPortPartnerOperSystemPriority | dot3adAggPortPartnerOperSystemPriority |
| 7.3.2.1.8 aAggPortPartnerAdminSystemID | dot3adAggPortPartnerAdminSystemID |
| 7.3.2.1.9 aAggPortPartnerOperSystemID | dot3adAggPortPartnerOperSystemID |
| 7.3.2.1.10 aAggPortPartnerAdminKey | dot3adAggPortPartnerAdminKey |
| 7.3.2.1.11 aAggPortPartnerOperKey | dot3adAggPortPartnerOperKey |
| 7.3.2.1.12 aAggPortSelectedAggID | dot3adAggPortSelectedAggID |
| 7.3.2.1.13 aAggPortAttachedAggID | dot3adAggPortAttachedAggID |
| 7.3.2.1.14 aAggPortActorPort | dot3adAggPortActorPort |
| 7.3.2.1.15 aAggPortActorPortPriority | dot3adAggPortActorPortPriority |
| 7.3.2.1.16 aAggPortPartnerAdminPort | dot3adAggPortPartnerAdminPort |
| 7.3.2.1.17 aAggPortPartnerOperPort | dot3adAggPortPartnerOperPort |
| 7.3.2.1.18 aAggPortPartnerAdminPortPriority | dot3adAggPortPartnerAdminPortPriority |
| 7.3.2.1.19 aAggPortPartnerOperPortPriority | dot3adAggPortPartnerOperPortPriority |
| 7.3.2.1.20 aAggPortActorAdminState | dot3adAggPortActorAdminState |
| 7.3.2.1.21 aAggPortActorOperState | dot3adAggPortActorOperState |
| 7.3.2.1.22 aAggPortPartnerAdminState | dot3adAggPortPartnerAdminState |
| 7.3.2.1.23 aAggPortPartnerOperState | dot3adAggPortPartnerOperState |
| 7.3.2.1.24 aAggPortAggregateOrIndividual | dot3adAggPortAggregateOrIndividual |
| 7.3.2.1.25 aAggPortOperConversationPasses | dot3adAggPortOperConversationPasses |
| 7.3.2.1.26 aAggPortOperConversationCollected | dot3adAggPortOperConversationCollected |
| 7.3.2.1.27 aAggPortAdminLinkNumber | dot3adAggPortLinkNumberId |
| -- deprecated -- | dot3adAggPortPartnerAdminLinkNumberId |
| 7.3.2.1.30 aAggPortWtrTime | dot3adAggPortWTRTime2 |
| -- deprecated -- | dot3adAggPortEnableLongPDUXmit |
| 7.3.2.1.28 aAggPortOperLinkNumber | dot3adAggPortOperLinkNumber |
| 7.3.2.1.29 aAggPortPartnerLinkNumber | dot3adAggPortPartnerLinkNumber |
| 7.3.2.1.31 aAggPortWtrRevertive | dot3adAggPortWtrRevertive |
| 7.3.2.1.32 aAggPortWtrWaiting | dot3adAggPortWtrWaiting |
| 7.3.2.1.33 aAggPortActorLacpVersion | dot3adAggPortActorLacpVersion |

**Table D.1—Managed object cross reference table *(continued)***

| Definition in Clause 7 | MIB object |
|---|---|
| 7.3.2.1.34 aAggPortPartnerLacpVersion | dot3adAggPortPartnerLacpVersion |
| 7.3.2.2.1 aAggPortProtocolDA | dot3adAggPortProtocolDA |
| 7.3.3.1.1 aAggPortStatsID | ifIndex value |
| 7.3.3.1.2 aAggPortStatsLACPDUsRx | dot3adAggPortStatsLACPDUsRx |
| 7.3.3.1.3 aAggPortStatsMarkerPDUsRx | dot3adAggPortStatsMarkerPDUsRx |
| 7.3.3.1.4 aAggPortStatsMarkerResponsePDUsRx | dot3adAggPortStatsMarkerResponsePDUsRx |
| 7.3.3.1.5 aAggPortStatsUnknownRx | dot3adAggPortStatsUnknownRx |
| 7.3.3.1.6 aAggPortStatsIllegalRx | dot3adAggPortStatsIllegalRx |
| 7.3.3.1.7 aAggPortStatsLACPDUsTx | dot3adAggPortStatsLACPDUsTx |
| 7.3.3.1.8 aAggPortStatsMarkerPDUsTx | dot3adAggPortStatsMarkerPDUsTx |
| 7.3.3.1.9 aAggPortStatsMarkerResponsePDUsTx | dot3adAggPortStatsMarkerResponsePDUsTx |
| 7.3.4.1.1 aAggPortDebugInformationID | ifIndex value (see IETF RFC 2863) of the port |
| 7.3.4.1.2 aAggPortDebugRxState | dot3adAggPortDebugRxState |
| 7.3.4.1.3 aAggPortDebugLastRxTime | dot3adAggPortDebugLastRxTime |
| 7.3.4.1.4 aAggPortDebugMuxState | dot3adAggPortDebugMuxState |
| 7.3.4.1.5 aAggPortDebugMuxReason | dot3adAggPortDebugMuxReason |
| -- deprecated -- | dot3adAggPortDebugActorChurnState |
| -- deprecated -- | dot3adAggPortDebugPartnerChurnState |
| -- deprecated -- | dot3adAggPortDebugActorChurnCount |
| -- deprecated -- | dot3adAggPortDebugPartnerChurnCount |
| 7.3.4.1.6 aAggPortDebugActorSyncTransitionCount | dot3adAggPortDebugActorSyncTransitionCount |
| 7.3.4.1.7 aAggPortDebugPartnerSyncTransitionCount | dot3adAggPortDebugPartnerSyncTransitionCount |
| 7.3.4.1.8 aAggPortDebugActorChangeCount | dot3adAggPortDebugActorChangeCount |
| 7.3.4.1.9 aAggPortDebugPartnerChangeCount | dot3adAggPortDebugPartnerChangeCount |
| -- deprecated -- | dot3adAggPortDebugActorCDSChurnState |
| -- deprecated -- | dot3adAggPortDebugPartnerCDSChurnState |
| -- deprecated -- | dot3adAggPortDebugActorCDSChurnCount |
| -- deprecated -- | dot3adAggPortDebugPartnerCDSChurnCount |
| 7.4.1.1.1 aDrniID | dot3adDrniIndex (ifIndex value) |
| 7.4.1.1.2 aDrniDescription | dot3adDrniDescription |
| 7.4.1.1.3 aDrniName | dot3adDrniName |
| -- obsolete -- | dot3adDrniPortalAddr |
| -- obsolete -- | dot3adDrniPortalPriority |

**Table D.1—Managed object cross reference table *(continued)***

| Definition in Clause 7 | MIB object |
|---|---|
| -- obsolete -- | dot3adDrniThreePortalSystem |
| -- obsolete -- | dot3adDrniPortalSystemNumber |
| -- obsolete -- | dot3adDrniIntraPortalLinkList |
| 7.4.1.1.4 aDrniAggregator | dot3adDrniAggregator |
| -- obsolete -- | dot3adDrniConvAdminGatewayTable |
| -- obsolete -- | dot3adDrniNeighborAdminConvGatewayListDigest |
| -- obsolete -- | dot3adDrniNeighborAdminConvPortListDigest |
| 7.4.1.1.5 aDrniAggregationPortList | dot3adDrniAggregationPortList |
| 7.4.1.1.6 aDrniHomeGatewayAlgorithm | dot3adDrniGatewayAlgorithm |
| -- obsolete -- | dot3adDrniNeighborAdminGatewayAlgorithm |
| -- obsolete -- | dot3adDrniNeighborAdminPortAlgorithm |
| -- obsolete -- | dot3adDrniNeighborAdminDRCPState |
| -- obsolete -- | dot3adDrniEncapsulationMethod |
| -- obsolete -- | dot3adDrniIPLEncapMapTable |
| -- obsolete -- | dot3adDrniNetEncapMapTable |
| 7.4.1.1.7 aDrniHomeAggregatorConversationPasses | dot3adDrniDRPortConversationPasses |
| 7.4.1.1.8 aDrniHomeGatewayConversationPasses | dot3adDrniDRGatewayConversationPasses |
| -- obsolete -- | dot3adDrniPSI |
| -- obsolete -- | dot3adDrniPortConversationControl |
| -- obsolete -- | dot3adDrniIntraPortalPortProtocolDA |
| 7.4.1.1.12 aDrniProtocolDA | dot3adDrniProtocolDA |
| 7.4.1.1.13 aDrniAggregatorSystem | dot3adDrniAggregatorSystem |
| 7.4.1.1.14 aDrniAggregatorSystemPriority | dot3adDrniAggregatorSystemPriority |
| 7.4.1.1.15 aDrniAggregatorKey | dot3adDrniAggregatorKey |
| 7.4.1.1.16 aDrniHomeCscdGatewayControl | dot3adDrniCscdGatewayControl |
| 7.4.1.1.17 aDrniHomeDRClientGatewayControl | dot3adDrniDRClientGatewayControl |
| 7.4.1.1.18 aDrniHomeGatewayEnableMask | dot3adDrniGatewayEnableMask |
| 7.4.1.1.19 aDrniHomeGatewayPreferenceMask | dot3adDrniGatewayPreferenceMask |
| 7.4.1.1.20 aDrniHomeAdminGatewayConvServiceMap | dot3adDrniAdminGatewayConvServiceMap |
| 7.4.1.1.21 aDrniHomeGatewayConvServiceDigest | dot3adDrniGatewayConvServiceDigest |
| 7.4.1.1.22 aDrniHomeGatewayAvailableMask | dot3adDrniGatewayAvailableMask |
| 7.4.1.1.23 aDrniIntraRelayPort | dot3adDrniIntraRelayPort |
| 7.4.1.1.24 aDrniHomeAdminIrpState | dot3adDrniHomeAdminIrpState |

**Table D.1—Managed object cross reference table *(continued)***

| Definition in Clause 7 | MIB object |
|---|---|
| 7.4.1.1.25 aDrniHomeOperIrpState | dot3adDrniHomeOperIrpState |
| 7.4.1.1.26 aDrniNeighborOperIrpState | dot3adDrniNeighborOperIrpState |
| 7.4.1.1.27 aDrniNeighborAggregatorConversationPasses | dot3adDrniNeighborPortConversationPasses |
| 7.4.1.1.28 aDrniNeighborGatewayConversationPasses | dot3adDrniNeighborGatewayConversationPasses |
| 7.4.1.1.29 aDrniNeighborSystem | dot3adDrniNeighborSystem |
| 7.4.1.1.30 aDrniNeighborSystemPriority | dot3adDrniNeighborSystemPriority |
| 7.4.1.1.31 aDrniNeighborDrniKey | dot3adDrniNeighborDrniKey |
| 7.4.1.1.32 aDrniSequenceNumbers | dot3adDrniSequenceNumbers |
| 7.4.1.1.33 aDrniNeighborAggregatorAlgorithm | dot3adDrniNeighborAggregatorAlgorithm |
| 7.4.1.1.34 aDrniNeighborAggregatorConvServiceDigest | dot3adDrniNeighborAggregatorConvServiceDigest |
| 7.4.1.1.35 aDrniNeighborAggregatorConvLinkDigest | dot3adDrniNeighborAggregatorConvLinkDigest |
| 7.4.1.1.36 aDrniNeighborPartnerSystemPriority | dot3adDrniNeighborPartnerSystemPriority |
| 7.4.1.1.37 aDrniNeighborPartnerSystem | dot3adDrniNeighborPartnerSystem |
| 7.4.1.1.38 aDrniNeighborPartnerAggregatorKey | dot3adDrniNeighborPartnerAggregatorKey |
| 7.4.1.1.39 aDrniNeighborCscdState | dot3adDrniNeighborCscdState |
| 7.4.1.1.40 aDrniNeighborActiveLinks | dot3adDrniNeighborActiveLinks |
| 7.4.1.1.41 aDrniNeighborGatewayAlgorithm | dot3adDrniNeighborGatewayAlgorithm |
| 7.4.1.1.42 aDrniNeighborGatewayConvServiceDigest | dot3adDrniNeighborGatewayConvServiceDigest |
| 7.4.1.1.43 aDrniNeighborGatewayAvailableMask | dot3adDrniNeighborGatewayAvailableMask |
| 7.4.1.1.44 aDrniNeighborGatewayPreferenceMask | dot3adDrniNeighborGatewayPreferenceMask |
| 7.4.1.1.45 aDrniDRCPDUsRx | dot3adDrniDRCPDUsRx |
| 7.4.1.1.46 aDrniIllegalRx | dot3adDrniIllegalRx |
| 7.4.1.1.47 aDrniDRCPDUsTx | dot3adIDrniDRCPDUsTx |
| -- obsolete -- | dot3adIPPPortConversationPasses |
| -- obsolete -- | dot3adIPPGatewayConversationDirection |
| -- obsolete -- | dot3adIPPAdminState |
| -- obsolete -- | dot3adIPPOperState |
| -- obsolete -- | dot3adIPPTimeOfLastOperChange |
| -- obsolete -- | dot3adIPPID (ifIndex value) |
| -- obsolete -- | dot3adIPPStatsDRCPDUsRx |
| -- obsolete -- | dot3adIPPStatsIllegalRx |
| -- obsolete -- | dot3adIPPStatsDRCPDUsTx |
| -- obsolete -- | ifIndex value |

**Table D.1—Managed object cross reference table *(continued)***

| Definition in Clause 7 | MIB object |
|---|---|
| -- obsolete -- | dot3adIPPDebugDRCPRxState |
| -- obsolete -- | dot3adIPPDebugLastRxTime |
| -- obsolete -- | dot3adIPPDebugDifferPortalReason |

**Table D.2—ifTable element definitions for an Aggregator**

| Object | Definition |
|---|---|
| ifIndex | A unique integer value is allocated to each Aggregator by the local System. Interpreted as defined in IETF RFC 2863. |
| ifDescr | Interpreted as defined in IETF RFC 2863 and as further refined in the definition of aAggDescription (7.3.1.1.2). |
| ifType | ieee8023adLag(161)[a]. |
| ifMTU | The largest MAC Client SDU that can be carried by this Aggregator—1500 octets. |
| ifSpeed | The data rate of the Aggregation as defined for aAggDataRate (7.3.1.1.16). |
| ifPhysAddress | The individual MAC Address of the Aggregator as defined for aAggMACAddress (7.3.1.1.9). |
| ifAdminStatus | The administrative state of the Aggregator as defined for aAggAdminState (7.3.1.1.13). |
| ifOperStatus | The operational state of the Aggregator as defined for aAggOperState (7.3.1.1.14). |
| ifLastChange | Interpreted as defined in IETF RFC 2863; see also the definition of aAggTimeOfLastOperChange (7.3.1.1.15). |
| ifInOctets | The total number of user data octets received by the aggregation, as defined for aAggOctetsRxOK (7.3.1.1.18). |
| ifInUcastPkts | The total number of unicast user data frames received by the aggregation. This value is calculated as the value of aAggFramesRxOK (7.3.1.1.20), less the values of aAggMulticastFramesRxOK (7.3.1.1.22) and aAggBroadcastFramesRxOK (7.3.1.1.24). |
| ifInNUcastPkts | Deprecated in IETF RFC 2863. |
| ifInDiscards | The number of frames discarded on reception, as defined for aAggFramesDiscardedOnRx (7.3.1.1.26). |
| ifInErrors | The number of frames with reception errors, as defined for aAggFramesWithRxErrors (7.3.1.1.28). |
| ifInUnknownProtos | The number of unknown protocol frames discarded on reception, as defined for aAggUnknownProtocolFrames (7.3.1.1.29). |
| ifOutOctets | The total number of user data octets transmitted by the aggregation, as defined for aAggOctetsTxOK (7.3.1.1.17). |
| ifOutUcastPkts | The total number of unicast user data frames transmitted by the aggregation. This value is calculated as the value of aAggFramesTxOK (7.3.1.1.19), less the values of aAggMulticastFramesTxOK (7.3.1.1.21) and aAggBroadcastFramesTxOK (7.3.1.1.23). |

**Table D.2—ifTable element definitions for an Aggregator *(continued)***

| Object | Definition |
|---|---|
| ifOutNUcastPkts | Deprecated in IETF RFC 2863. |
| ifOutDiscards | The number of frames discarded on transmission, as defined for aAggFramesDiscardedOnTx (7.3.1.1.25). |
| ifOutErrors | The number of frames discarded due to transmission errors, as defined for aAggFramesWithTxErrors (7.3.1.1.27). |
| ifOutQLen | Deprecated in IETF RFC 2863. Set to zero if present. |
| ifSpecific | Deprecated in IETF RFC 2863. Set to { 0.0 } if present. |
| ifLinkUpDownTrapEnable | See the definition of aAggLinkUpDownNotificationEnable (7.3.1.1.31). |
| ifConnectorPresent | "FALSE." |
| ifHighSpeed | Set to zero. |
| ifName | The locally assigned textual name of the Aggregator, as defined for aAggName (7.3.1.1.3). Interpreted as defined in IETF RFC 2863. |
| linkUp TRAP | See the definition of nAggLinkUpNotification (7.3.1.2.1). |
| linkDown TRAP | See the definition of nAggLinkDownNotification (7.3.1.2.2). |

[a] Values of ifType are assigned by the Internet Assigned Numbers Authority (IANA). A directory of number assignments is maintained on their website, at URL: http://www.iana.org/numbers.html. The currently assigned ifType values can be found in the SMI Numbers (Network Management Parameters) section of that directory.

**Table D.3—ifTable element definitions for a DRNI Gateway**

| Object | Definition |
|---|---|
| ifIndex | A unique integer value is allocated to each DRNI Gateway by the local System. Interpreted as defined in IETF RFC 2863. |
| ifDescr | Interpreted as defined in IETF RFC 2863 and as further refined in the definition of aDrniDescription (7.4.1.1.2). |
| ifType | ieee8021axDrni(297)[a]. |
| ifMTU | Set to the same value as the Aggregator supporting this DRNI Gateway. |
| ifSpeed | Set to zero. |
| ifPhysAddress | The individual MAC Address of the Aggregator supporting this DRNI Gateway as defined in aAggMACAddress (7.3.1.1.9). |
| ifAdminStatus | The administrative state of the DRNI Gateway as defined for aDrniGatewayAdminState (7.4.1.1.9). |
| ifOperStatus | The operational state of the DRNI Gateway as defined for aDrniGatewayOperState (7.4.1.1.10). |
| ifLastChange | Interpreted as defined in IETF RFC 2863. |
| ifInOctets | As defined in IETF RFC 2863. |
| ifInUcastPkts | As defined in IETF RFC 2863. |
| ifInNUcastPkts | Deprecated in IETF RFC 2863. |

**Table D.3—ifTable element definitions for a DRNI Gateway *(continued)***

| Object | Definition |
|---|---|
| ifInDiscards | As defined in IETF RFC 2863. |
| ifInErrors | As defined in IETF RFC 2863. |
| ifInUnknownProtos | The number of unknown protocol frames discarded on reception, as defined for aDrniIllegalRx (7.4.1.1.46). |
| ifOutOctets | As defined in IETF RFC 2863. |
| ifOutUcastPkts | As defined in IETF RFC 2863. |
| ifOutNUcastPkts | Deprecated in IETF RFC 2863. |
| ifOutDiscards | As defined in IETF RFC 2863. |
| ifOutErrors | As defined in IETF RFC 2863. |
| ifOutQLen | Deprecated in IETF RFC 2863. Set to zero if present. |
| ifSpecific | Deprecated in IETF RFC 2863. Set to { 0.0 } if present. |
| ifLinkUpDownTrapEnable | Interpreted as defined in IETF RFC 2863. Set to disabled(2). |
| ifConnectorPresent | "FALSE." |
| ifHighSpeed | Set to zero. |
| ifName | The locally assigned textual name of the Aggregator, as defined for aDrniName (7.4.1.1.3). Interpreted as defined in IETF RFC 2863. |

## D.4.2 MIB Subtrees

The correspondence between the objects in the MIB module subtrees and the objects of the managed object classes defined in Clause 7 is described in the following subclauses.

### D.4.2.1 The dot3adAgg Subtree

The objects of the dot3adAgg subtree correspond to the objects of the Aggregator managed object class (7.3.1) with the exception of the Aggregator Notifications (7.3.1.1.37).

### D.4.2.2 The dot3adAggPort Subtree

The objects of the dot3adAggPort subtree correspond to the objects of the Aggregation Port managed object class (7.3.2), the Aggregation Port Statistics managed object class (7.3.3) and the Aggregation Port Debug Information managed object class (7.3.4).

### D.4.2.3 The dot3adAggNotifications Subtree

The objects of the dot3adAggPort subtree correspond to the Aggregator Notifications (7.3.1.1.37).

### D.4.2.4 The dot3adDrni Subtree

The objects of the dot3adDrni subtree correspond to the objects of the Distributed Relay Managed Object Class (7.4.1).

## D.5 Relationship to other MIBs

It is assumed that a System implementing this MIB will also implement (at least) the "system" group defined in MIB-II defined in IETF RFC 1213 and the "interfaces" group defined in IETF RFC 2863.

### D.5.1 Relationship to the Interfaces MIB

IETF RFC 2863, the Interface MIB Evolution, requires that any MIB that is an adjunct of the Interface MIB, clarify specific areas within the Interface MIB. These areas were intentionally left vague in IETF RFC 2863 to avoid over constraining the MIB, thereby precluding management of certain media types.

Section 3.3 of IETF RFC 2863 enumerates several areas that a media-specific MIB has to clarify. Each of these areas is addressed in D.5.2 and D.5.3. The implementer is referred to IETF RFC 2863 in order to understand the general intent of these areas.

In IETF RFC 2863, the "interfaces" group is defined as being mandatory for all Systems and contains information on an entity's interfaces, where each interface is thought of as being attached to a *subnetwork*. (Note that this term is not to be confused with *subnet,* which refers to an addressing partitioning scheme used in the Internet suite of protocols.) The term *segment* is sometimes used to refer to such a subnetwork.

Implicit in this MIB is the notion of Aggregators and Aggregation Ports. Each of these Aggregators and Aggregation Ports is associated with one interface of the "interfaces" group (one row in the ifTable) and each Aggregation Port is associated with a different interface.

Each Aggregator and Aggregation Port is uniquely identified by an interface number (ifIndex). The ifIndex value assigned to a given Aggregation Port is the same as the ifIndex value assigned to the MAC interface with which that Aggregation Port is associated.

### D.5.2 Layering model

This annex assumes the interpretation of the Interfaces Group to be in accordance with IETF RFC 2863, which states that the ifTable contains information on the managed resource's interfaces and that each sublayer below the internetwork layer of a network interface is considered an interface.

This annex recommends that within an entity, Aggregators and DRNI Gateways that are instantiated as an entry in dot3adAggTable are also represented by an entry in ifTable.

Where an entity contains Link Aggregation entities that transmit and receive traffic to/from an Aggregator or DRNI Gateway, these are represented in the ifTable as interfaces of type ieee8023adLag(161) or ieee8021axDrni(297).

### D.5.3 ifStackTable

If the ifStackTable defined in IETF RFC 2863 is implemented, then

a) The relationship shown in the table has the property that an Aggregator is a higher interface relative to an Aggregation Port, and a DRNI Gateway is a higher interface relative to an Aggregator.
b) This relationship is read-only.

NOTE—The restriction stated here is intended to enforce a strict hierarchical relationship between Aggregations and Aggregation Ports, and to prevent those relationships from being modified. The read-only restriction does not apply to any other relationships that can be expressed in the ifStackTable.

### D.5.4 ifRcvAddressTable

The ifRcvAddressTable contains all MAC Addresses, unicast, multicast, and broadcast, for which an interface can receive frames and forward them up to a higher layer entity for local consumption. An Aggregator has at least one such address.

## D.6 Definitions for Link Aggregation MIB

In the following MIB definition,[12] any discrepancy between the DESCRIPTION text and the BEHAVIOR DEFINED AS in the corresponding definition in Clause 7, the definition in Clause 7 shall take precedence.

NOTE 1—There is no requirement for persistency of the objects in the implementations of this MIB module. No guidance is provided for the discontinuity of any counters in the MIB module.

NOTE 2—Many managed objects do not have DEFVAL values. Link Aggregation operation is designed to be functional with any value of these parameters.



[12] MIB definitions are available at http://www.ieee802.org/1/pages/MIBS.html.

```
IEEE8023-LAG-MIB DEFINITIONS ::= BEGIN

-- ------------------------------------------------------------
-- IEEE 802.1AX MIB
-- ------------------------------------------------------------


IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, Counter64, Integer32,
            TimeTicks, NOTIFICATION-TYPE
        FROM SNMPv2-SMI
    DisplayString, MacAddress, TEXTUAL-CONVENTION, TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF
    InterfaceIndex
        FROM IF-MIB
    PortList
        FROM Q-BRIDGE-MIB
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    ;


lagMIB MODULE-IDENTITY
    LAST-UPDATED "201908210000Z"
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
        "  WG-URL: http://www.ieee802.org/1/index.html
         WG-EMail: stds-802-1@ieee.org

          Contact: IEEE 802.1 Working Group Chair
           Postal: C/O IEEE 802.1 Working Group
                   IEEE Standards Association
                   445 Hoes Lane
                   P.O. Box 1331
                   Piscataway
                   NJ 08855-1331
                   USA
           E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
    DESCRIPTION
        "The Link Aggregation module for 802.1AX-Rev-2020."
    REVISION "201908210000Z"
    DESCRIPTION
        "The Link Aggregation module for managing IEEE 802.1AX-2014 as
        updated by 802.1AX-2014-Cor1."
    REVISION "201610120000Z"
    DESCRIPTION
        "The Link Aggregation module for managing IEEE 802.1AX-REV."
    REVISION "201412180000Z"
    DESCRIPTION
        "The Link Aggregation module for managing IEEE 802.1AX."
    REVISION "201201160000Z"
    DESCRIPTION
```

```
        "Updated for IEEE 802.1AXbk"
    REVISION "200706290000Z"
    DESCRIPTION
        "References updated 04 Jun 2007 for IEEE 802.1AX"
    REVISION "200006270000Z"
    DESCRIPTION
        "Original publication IEEE 802.3ad"
    ::= { iso(1) member-body(2) us(840) ieee802dot3(10006) snmpmibs(300)
43 }


-- ----------------------------------------------------------
-- Textual Conventions
-- ----------------------------------------------------------


LacpKey ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS       current
    DESCRIPTION
        "The Actor or Partner Key value."
    SYNTAX      Integer32 (0..65535)

LacpState ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The Actor and Partner State values from the LACPDU."
    REFERENCE
        "Figure 6-9"
    SYNTAX      BITS {
                    lacpActivity(0),
                    lacpTimeout(1),
                    aggregation(2),
                    synchronization(3),
                    collecting(4),
                    distributing(5),
                    defaulted(6),
                    expired(7)
                }

IrpState ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The Intra-Relay Port State values from the DRCPDU (9.6.2.3)."
    REFERENCE
        "Figure 9-13"
    SYNTAX      BITS {
                    distributedRelayNumberLSB(0),
                    distributedRelayNumberMSB(1),
                    shortTimeout(2),
                    synchronization(3),
                    ircData(4),
                    drni(5),
                    defaulted(6),
```

```
                    expired(7)
                }

CscdState ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The CSCD State values from the Aggregator State TLV (9.6.2.4)."
    REFERENCE
        "Figure 9-15"
    SYNTAX      BITS {
                    reserved0(0),
                    reserved1(1),
                    reserved2(2),
                    cscdGatewayControl(3),
                    discardWrongConversation(4),
                    differConvLinkDigests(5),
                    differConvServiceDigests(6),
                    differPortAlgorithms(7)
                }

DrcpState ::= TEXTUAL-CONVENTION
    STATUS      obsolete
    DESCRIPTION
        "Administrative values of DRCP state."
    SYNTAX      BITS {
                    homeGateway(0),
                    neighborGateway(1),
                    otherGateway(2),
                    ippActivity(3),
                    timeout(4),
                    gatewaySync(5),
                    portSync(6),
                    expired(7)
                }

ChurnState ::= TEXTUAL-CONVENTION
    STATUS      deprecated
    DESCRIPTION
        "The state of the Churn Detection machine."
    SYNTAX      INTEGER {
                    noChurn(1),
                    churn(2),
                    churnMonitor(3) -- deprecated
}

AggState ::= TEXTUAL-CONVENTION
    STATUS     current
    DESCRIPTION
        "The state of the object entry."
    SYNTAX     INTEGER {
                    up(1),
                    down(2)
                }
```

```
DrniConvAdminGatewayList ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "1x,"
    STATUS      obsolete
    DESCRIPTION
        "The three elements of the octet string represent the
        three portal system numbers in order of priority with
        highest priority first."
    SYNTAX      OCTET STRING (SIZE (3))

PortalLinkList ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "4d,"
    STATUS      obsolete
    DESCRIPTION
        "Each four octets of the octet string represent an
        ifIndex for an Intra-Port Link. The first ifIndex is
        to Portal System 1, the second ifIndex is to Portal
        System 2 and the third ifIndex is to portal System 3.
        The ifIndex of the current portal system is set to zero."
    SYNTAX      OCTET STRING (SIZE (12))

ServiceIdList ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "4d,"
    STATUS      current
    DESCRIPTION
        "A list which contains, in general, a set of 32-bit values
        interpreted as Service Identifiers. An empty set is represented
        as an octet string of size zero."
    SYNTAX      OCTET STRING

-- ----------------------------------------------------------
-- subtrees in the LAG MIB
-- ----------------------------------------------------------

lagMIBNotifications OBJECT IDENTIFIER   ::= { lagMIB 0 }
lagMIBObjects OBJECT IDENTIFIER         ::= { lagMIB 1 }
dot3adAggConformance OBJECT IDENTIFIER  ::= { lagMIB 2 }

dot3adAgg OBJECT IDENTIFIER             ::= { lagMIBObjects 1 }
dot3adAggPort OBJECT IDENTIFIER         ::= { lagMIBObjects 2 }
dot3adDrni OBJECT IDENTIFIER            ::= { lagMIBObjects 4 }
dot3adIPP OBJECT IDENTIFIER             ::= { lagMIBObjects 5 }

-- ----------------------------------------------------------
-- The Tables Last Changed Object
-- ----------------------------------------------------------

dot3adTablesLastChanged OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object indicates the time of the
        most recent change to the dot3adAggTable,
        dot3adAggPortTable, dot3adDrniTable and
```

```
        dot3adIPPAttributeTable."

::= { lagMIBObjects 3 }

-- ----------------------------------------------------------------
-- The Aggregator Configuration Table
-- ----------------------------------------------------------------

dot3adAggTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains information about every
        Aggregator that is associated with this System."
    REFERENCE
        "7.3.1"
    ::= { dot3adAgg 1 }

dot3adAggEntry OBJECT-TYPE
    SYNTAX      Dot3adAggEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of the Aggregator parameters. This is indexed
        by the ifIndex of the Aggregator."
    INDEX { dot3adAggIndex }
    ::= { dot3adAggTable 1 }

Dot3adAggEntry ::=
    SEQUENCE {
        dot3adAggIndex
            InterfaceIndex,
        dot3adAggMACAddress
            MacAddress,
        dot3adAggActorSystemPriority
            Integer32,
        dot3adAggActorSystemID
            MacAddress,
        dot3adAggAggregateOrIndividual
            TruthValue,
        dot3adAggActorAdminKey
            LacpKey,
        dot3adAggActorOperKey
            LacpKey,
        dot3adAggPartnerSystemID
            MacAddress,
        dot3adAggPartnerSystemPriority
            Integer32,
        dot3adAggPartnerOperKey
            LacpKey,
        dot3adAggCollectorMaxDelay
            Integer32
    }
```

```
dot3adAggIndex OBJECT-TYPE
    SYNTAX       InterfaceIndex
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The unique identifier allocated to this Aggregator by the
        local System. This attribute identifies an Aggregator instance
        among the subordinate managed objects of the containing object.
        This value is read-only. NOTE-The aAggID is represented in the
        SMIv2 MIB as an ifIndex-see D.4.1."
    REFERENCE
        "7.3.1.1.1"
    ::= { dot3adAggEntry 1 }


dot3adAggMACAddress OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 6-octet read-only value carrying the individual
        MAC address assigned to the Aggregator."
    REFERENCE
        "7.3.1.1.9"
    ::= { dot3adAggEntry 2 }


dot3adAggActorSystemPriority OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A 2-octet read-write value indicating the priority value
        associated with the Actor's System Identifier."
    REFERENCE
        "7.3.1.1.5"
    ::= { dot3adAggEntry 3 }


dot3adAggActorSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A 6-octet read-write MAC address value used as a unique
        identifier for the System that contains this Aggregator.
        NOTE-From the perspective of the Link Aggregation
        mechanisms described in Clause 6, only a single
        combination of Actor's System ID and System Priority are
        considered, and no distinction is made between the
        values of these parameters for an Aggregator and the
        port(s) that are associated with it; i.e., the protocol
        is described in terms of the operation of aggregation
```

```
        within a single System. However, the managed objects
        provided for the Aggregator and the port both allow
        management of these parameters. The result of this is to
        permit a single piece of equipment to be configured by
        management to contain more than one System from the
        point of view of the operation of Link Aggregation. This
        can be of particular use in the configuration of
        equipment that has limited aggregation capability (see 6.7)."
    REFERENCE
        "7.3.1.1.4"
    ::= { dot3adAggEntry 4 }


dot3adAggAggregateOrIndividual OBJECT-TYPE
    SYNTAX       TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A read-only Boolean value indicating whether the
        Aggregator represents an Aggregate ('TRUE') or
        an Individual link ('FALSE')."
    REFERENCE
        "7.3.1.1.6"
    ::= { dot3adAggEntry 5 }


dot3adAggActorAdminKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The current administrative value of the Key for the
        Aggregator. The administrative Key value can differ from the
        operational Key value for the reasons discussed in 6.7.2. This
        is a 16-bit read-write value. The meaning of particular Key
        values is of local significance."
    REFERENCE
        "7.3.1.1.7"
    ::= { dot3adAggEntry 6 }


dot3adAggActorOperKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current operational value of the Key for the Aggregator.
        The administrative Key value can differ from the operational
        Key value for the reasons discussed in 6.7.2. This is a 16-bit
        read-only value. The meaning of particular Key values is of
        local significance."
    REFERENCE
        "7.3.1.1.8"
    ::= { dot3adAggEntry 7 }
```

```
dot3adAggPartnerSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 6-octet read-only MAC address value consisting of the
        unique identifier for the current protocol Partner of this
        Aggregator. A value of zero indicates that there is no known
        Partner. If the aggregation is manually configured, this
        System ID value will be a value assigned by the local System."
    REFERENCE
        "7.3.1.1.10"
    ::= { dot3adAggEntry 8 }


dot3adAggPartnerSystemPriority OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 2-octet read-only value that indicates the priority
        value associated with the Partner's System ID. If the
        aggregation is manually configured, this System Priority value
        will be a value assigned by the local System."
    REFERENCE
        "7.3.1.1.11"
    ::= { dot3adAggEntry 9 }


dot3adAggPartnerOperKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current operational value of the Key for the
        Aggregator's current protocol Partner. This is a 16-bit
        read-only value. If the aggregation is manually configured,
        this Key value will be a value assigned by the local System."
    REFERENCE
        "7.3.1.1.12"
    ::= { dot3adAggEntry 10 }


dot3adAggCollectorMaxDelay OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The value of this 16-bit read-write attribute defines
        the maximum delay, in tens of microseconds, that
        can be imposed by the Frame Collector between
        receiving a frame from an Aggregator Parser, and
```

```
        either delivering the frame to its Aggregator Client
        or discarding the frame (see 6.2.3.1.1)."
    REFERENCE
        "7.3.1.1.32"
    ::= { dot3adAggEntry 11 }

-- ----------------------------------------------------------
-- The Aggregation Port List Table
-- ----------------------------------------------------------



dot3adAggPortListTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggPortListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains a list of all the ports
        associated with each Aggregator."
    REFERENCE
        "7.3.1.1.30"
    ::= { dot3adAgg 2 }

dot3adAggPortListEntry OBJECT-TYPE
    SYNTAX      Dot3adAggPortListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of the ports associated with a given Aggregator.
        This is indexed by the ifIndex of the Aggregator."
    INDEX { dot3adAggIndex }
    ::= { dot3adAggPortListTable 1 }

Dot3adAggPortListEntry ::=
    SEQUENCE {
        dot3adAggPortListPorts
            PortList
    }

dot3adAggPortListPorts OBJECT-TYPE
    SYNTAX      PortList
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The complete set of ports currently associated with
        this Aggregator.  Each bit set in this list represents
        an Actor Port member of this Link Aggregation."
    REFERENCE
        "7.3.1.1.30"
    ::= { dot3adAggPortListEntry 1 }

-- ----------------------------------------------------------
-- The Aggregation Extension Table
-- ----------------------------------------------------------
```

```
dot3adAggXTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggXEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that extends dot3adAggTable."
    REFERENCE
        "7.3.1.1"
    ::= { dot3adAgg 3 }

dot3adAggXEntry OBJECT-TYPE
    SYNTAX      Dot3adAggXEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of extension parameters for the Aggregator
        Configuration Table"
    AUGMENTS { dot3adAggEntry }
    ::= { dot3adAggXTable 1 }

Dot3adAggXEntry ::=
    SEQUENCE {
        dot3adAggDescription
            DisplayString,
        dot3adAggName
            DisplayString,
        dot3adAggAdminState
            AggState,
        dot3adAggOperState
            AggState,
        dot3adAggTimeOfLastOperChange
            Integer32,
        dot3adAggDataRate
            Integer32,
        dot3adAggOctetsTxOK
            Counter64,
        dot3adAggOctetsRxOK
            Counter64,
        dot3adAggFramesTxOK
            Counter64,
        dot3adAggFramesRxOK
            Counter64,
        dot3adAggMulticastFramesTxOK
            Counter64,
        dot3adAggMulticastFramesRxOK
            Counter64,
        dot3adAggBroadcastFramesTxOK
            Counter64,
        dot3adAggBroadcastFramesRxOK
            Counter64,
        dot3adAggFramesDiscardedOnTx
            Counter64,
        dot3adAggFramesDiscardedOnRx
```

```
            Counter64,
        dot3adAggFramesWithTxErrors
            Counter64,
        dot3adAggFramesWithRxErrors
            Counter64,
        dot3adAggUnknownProtocolFrames
            Counter64,
        dot3adAggLinkUpDownNotificationEnable
            Integer32,
        dot3adAggPortAlgorithm
            OCTET STRING,
        dot3adAggPartnerAdminPortAlgorithm
            OCTET STRING,
        dot3adAggPartnerAdminPortConversationListDigest
            OCTET STRING,
        dot3adAggAdminDiscardWrongConversation
            TruthValue,
        dot3adAggPartnerAdminConvServiceMappingDigest
            OCTET STRING,
        dot3adAggAdminDiscardWrongConversation2
            INTEGER,
        dot3adAggOperDiscardWrongConversation
            TruthValue,
        dot3adAggConvLinkDigest
            OCTET STRING,
        dot3adAggConvServiceDigest
            OCTET STRING,
        dot3adAggPartnerPortAlgorithm
            OCTET STRING,
        dot3adAggPartnerConvLinkDigest
            OCTET STRING,
        dot3adAggPartnerConvServiceDigest
            OCTET STRING,
        dot3adAggActiveLinks
            OCTET STRING
    }

dot3adAggDescription OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A human-readable text string containing information about
        the Aggregator. This string could include information about
        the distribution algorithm in use on this Aggregator; for
        example, 'Aggregator 1, Dist Alg=Dest MAC address.' This
        string is read-only. The contents are vendor specific."
    REFERENCE
        "7.3.1.1.2"
    ::= { dot3adAggXEntry 1 }

dot3adAggName OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-write
```

```
        STATUS       current
        DESCRIPTION
            "A human-readable text string containing a locally significant
            name for the Aggregator. This string is read-write."
        REFERENCE
            "7.3.1.1.3"
        ::= { dot3adAggXEntry 2 }

    dot3adAggAdminState OBJECT-TYPE
        SYNTAX       AggState
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "This read-write value defines the administrative state of
            the Aggregator. A value of 'up' indicates that the operational
            state of the Aggregator (aAggOperState) is permitted to be
            either up or down. A value of 'down' forces the operational
            state of the Aggregator to be down. Changes to the
            administrative state affect the operational state of the
            Aggregator only, not the operational state of the Aggregation
            Ports that are attached to the Aggregator. A GET operation
            returns the current administrative state. A SET operation
            changes the administrative state to a new value."
        REFERENCE
            "7.3.1.1.13"
        ::= { dot3adAggXEntry 3 }

    dot3adAggOperState OBJECT-TYPE
        SYNTAX       AggState
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "This read-only value defines the operational state of the
            Aggregator. An operational state of 'up' indicates that the
            Aggregator is available for use by the Aggregator Client;
            a value of 'down' indicates that the Aggregator is not
            available for use by the Aggregator Client."
        REFERENCE
            "7.3.1.1.14"
        ::= { dot3adAggXEntry 4 }

    dot3adAggTimeOfLastOperChange OBJECT-TYPE
        SYNTAX       Integer32
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The time at which the interface entered its current
            operational state, in terms of centiseconds since the
            system was last reset. If the current state was entered
            prior to the last reinitialization of the local network
            management subsystem, then this object contains a value of
            zero. The ifLastChange object in the Interfaces MIB defined
            in IETF RFC 2863 is a suitable object for supplying a value
            for aAggTimeOfLastOperChange. This value is read-only.
```

```
        Note - aAggTimeOfLastOperChange was defined in terms of
        the aTimeSinceSystemReset variable of IEEE Std 802.3-2018,
        F.2.1, in earlier versions of this standard.
        aTimeSinceSystemReset and ifLastChange have the same meaning."
    REFERENCE
        "7.3.1.1.15"
    ::= { dot3adAggXEntry 5 }

dot3adAggDataRate OBJECT-TYPE
    SYNTAX       Integer32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current data rate, in bits per second, of the aggregate
        link. The value is calculated as the sum of the data rate of
        each link in the aggregation. This attribute is read-only."
    REFERENCE
        "7.3.1.1.16"
    ::= { dot3adAggXEntry 6 }

dot3adAggOctetsTxOK OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "octets"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of the data and padding octets transmitted by this
        Aggregator on all Aggregation Ports that are (or have been)
        members of the aggregation. The count does not include octets
        transmitted by the Aggregator in frames that carry LACPDUs or
        Marker PDUs (7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9). However, it
        includes frames discarded by the Frame Distribution function of
        the Aggregator (7.3.1.1.25). This value is read-only."
    REFERENCE
        "7.3.1.1.17"
    ::= { dot3adAggXEntry 7 }

dot3adAggOctetsRxOK OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "octets"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of the data and padding octets received by this
        Aggregator, from the Aggregation Ports that are (or have been)
        members of the aggregation. The count does not include octets
        received in frames that carry LACP or Marker PDUs (7.3.3.1.2,
        7.3.3.1.3, 7.3.3.1.4), or frames discarded by the Frame
        Collection function of the Aggregator (7.3.1.1.26). This value
        is read-only."
    REFERENCE
        "7.3.1.1.18"
```

```
        ::= { dot3adAggXEntry 8 }

    dot3adAggFramesTxOK OBJECT-TYPE
        SYNTAX       Counter64
        UNITS
            "frames"
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A count of the data frames transmitted by this Aggregator on
            all Aggregation Ports that are (or have been) members of the
            aggregation. The count does not include frames transmitted by
            the Aggregator that carry LACP or Marker PDUs (7.3.3.1.7,
            7.3.3.1.8, 7.3.3.1.9). However, it includes frames discarded
            by the Frame Distribution function of the Aggregator
            (7.3.1.1.25). This value is read-only."
        REFERENCE
            "7.3.1.1.19"
        ::= { dot3adAggXEntry 9 }

    dot3adAggFramesRxOK OBJECT-TYPE
        SYNTAX       Counter64
        UNITS
            "frames"
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A count of the data frames received by this Aggregator, from
            the Aggregation Ports that are (or have been) members of the
            aggregation. The count does not include frames that carry LACP
            or Marker PDUs (7.3.3.1.2, 7.3.3.1.3, 7.3.3.1.4), or frames
            discarded by the Frame Collection function of the Aggregator
            (7.3.1.1.26). This value is read-only."
        REFERENCE
            "7.3.1.1.20"
        ::= { dot3adAggXEntry 10 }

    dot3adAggMulticastFramesTxOK OBJECT-TYPE
        SYNTAX       Counter64
        UNITS
            "frames"
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A count of the data frames transmitted by this Aggregator on
            all Aggregation Ports that are (or have been) members of the
            aggregation, to a group DA other than the broadcast address.
            The count does not include frames transmitted by the
            Aggregator that carry LACP or Marker PDUs (7.3.3.1.7,
            7.3.3.1.8, 7.3.3.1.9). However, it includes frames discarded
            by the Frame Distribution function of the Aggregator
            (7.3.1.1.25). This value is read-only."
        REFERENCE
            "7.3.1.1.21"
```

```
    ::= { dot3adAggXEntry 11 }

dot3adAggMulticastFramesRxOK OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "frames"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of the data frames received by this Aggregator, from
        the Aggregation Ports that are (or have been) members of the
        aggregation, that were addressed to an active group address,
        other than the broadcast address. The count does not include
        frames that carry LACP or Marker PDUs (7.3.3.1.2, 7.3.3.1.3,
        7.3.3.1.4), or frames discarded by the Frame Collection
        function of the Aggregator (7.3.1.1.26). This value is
        read-only."
    REFERENCE
        "7.3.1.1.22"
    ::= { dot3adAggXEntry 12 }

dot3adAggBroadcastFramesTxOK OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "frames"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of the broadcast data frames transmitted by this
        Aggregator on all Aggregation Ports that are (or have been)
        members of the aggregation. The count does not include frames
        transmitted by the Aggregator that carry LACP or Marker PDUs
        (7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9). However, it includes frames
        discarded by the Frame Distribution function of the Aggregator
        (7.3.1.1.25). This value is read-only."
    REFERENCE
        "7.3.1.1.23"
    ::= { dot3adAggXEntry 13 }

dot3adAggBroadcastFramesRxOK OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "frames"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of the broadcast data frames received by this
        Aggregator, from the Aggregation Ports that are (or have been)
        members of the aggregation. The count does not include frames
        that carry LACP or Marker PDUs (7.3.3.1.2, 7.3.3.1.3,
        7.3.3.1.4), illegal or unknown protocol frames (7.3.3.1.5,
        7.3.3.1.6), or frames discarded by the Frame Collection
        function of the Aggregator (7.3.1.1.26). This value is
        read-only."
```

```
        REFERENCE
            "7.3.1.1.24"
        ::= { dot3adAggXEntry 14 }

    dot3adAggFramesDiscardedOnTx OBJECT-TYPE
        SYNTAX       Counter64
        UNITS
            "frames"
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A count of data frames requested to be transmitted by this
            Aggregator that were discarded by the Frame Distribution
            function of the Aggregator when conversations are reallocated
            to different Aggregation Ports, due to the requirement to
            ensure that the conversations are flushed on the old
            Aggregation Ports in order to maintain proper frame ordering
            (B.3), or discarded as a result of excessive collisions by
            Aggregation Ports that are (or have been) members of the
            aggregation. This value is read-only."
        REFERENCE
            "7.3.1.1.25"
        ::= { dot3adAggXEntry 15 }

    dot3adAggFramesDiscardedOnRx OBJECT-TYPE
        SYNTAX       Counter64
        UNITS
            "frames"
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A count of data frames, received on all Aggregation Ports
            that are (or have been) members of the aggregation, that
            were discarded by the Frame Collection function of the
            Aggregator as they were received on Aggregation Ports whose
            Frame Collection function was disabled. This value is
            read-only."
        REFERENCE
            "7.3.1.1.26"
        ::= { dot3adAggXEntry 16 }

    dot3adAggFramesWithTxErrors OBJECT-TYPE
        SYNTAX       Counter64
        UNITS
            "frames"
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A count of data frames requested to be transmitted by this
            Aggregator that experienced transmission errors on Aggregation
            Ports that are (or have been) members of the aggregation. This
            count does not include frames discarded due to excess
            collisions. This value is read-only."
        REFERENCE
```

```
        "7.3.1.1.27"
    ::= { dot3adAggXEntry 17 }

dot3adAggFramesWithRxErrors OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "frames"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of data frames discarded on reception by all
        Aggregation Ports that are (or have been) members of the
        aggregation, or that were discarded by the Frame Collection
        function of the Aggregator, or that were discarded by the
        Aggregator due to the detection of an illegal Slow Protocols
        PDU (7.3.3.1.6). This value is read-only."
    REFERENCE
        "7.3.1.1.28"
    ::= { dot3adAggXEntry 18 }

dot3adAggUnknownProtocolFrames OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "frames"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A count of data frames discarded on reception by all
        Aggregation Ports that are (or have been) members of the
        aggregation, due to the detection of an unknown Slow Protocols
        PDU (7.3.3.1.5). This value is read-only."
    REFERENCE
        "7.3.1.1.29"
    ::= { dot3adAggXEntry 19 }

dot3adAggLinkUpDownNotificationEnable OBJECT-TYPE
    SYNTAX       INTEGER {
                     enabled(1),
                     disabled(2)
                 }
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "When set to 'enabled', Link Up and Link Down notifications are
        enabled for this Aggregator. When set to 'disabled', Link Up
        and Link Down notifications are disabled for this Aggregator.
        This value is read-write."
    REFERENCE
        "7.3.1.1.31"
    ::= { dot3adAggXEntry 20 }

dot3adAggPortAlgorithm OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (4))
    MAX-ACCESS  read-write
```

```
        STATUS      current
        DESCRIPTION
            "This object identifies the algorithm used by the Aggregator to
            assign frames to a Port Conversation ID. Table 6-4 provides the
            IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings. A SEQUENCE
            OF OCTETS consisting of a 3-octet OUI or CID and one
            following octet."
        REFERENCE
            "7.3.1.1.33"
        ::= { dot3adAggXEntry 21 }

    dot3adAggPartnerAdminPortAlgorithm OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE (4))
        MAX-ACCESS  read-write
        STATUS      deprecated
        DESCRIPTION
            "This object identifies the value for the algorithm of the
            Partner System, administratively assigned
            for use when the Partner's information is unknown. Table 6-4
            provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm
            encodings. Its default value is set to the
            'Unspecified' value in Table 6-4. A SEQUENCE OF OCTETS
            consisting of a 3-octet OUI or CID and one following octet.
            This parameter was eliminated because there was no reason for
            it ever to have a value different from the default value."
        REFERENCE
            "7.3.1.1.34 of IEEE Std 802.1AX-2014"
        DEFVAL  { '0080C200'H }
        ::= { dot3adAggXEntry 22 }

    dot3adAggPartnerAdminPortConversationListDigest OBJECT-TYPE
        SYNTAX       OCTET STRING (SIZE (16))
        MAX-ACCESS read-write
        STATUS       deprecated
        DESCRIPTION
            "The value for the digest of the prioritized Port Conversation
            ID-to-Link Number assignments of the Partner System,
            administratively assigned for use when the Partner's
            information is unknown. Its default value is set to NULL.
            This parameter was eliminated because there was no reason for
            it ever to have a value different from the default value."
        REFERENCE
            "7.3.1.1.36 of IEEE Std 802.1AX-2014"
        DEFVAL  { ''H }
        ::= { dot3adAggXEntry 23 }

    dot3adAggAdminDiscardWrongConversation OBJECT-TYPE
        SYNTAX       TruthValue
        MAX-ACCESS   read-write
        STATUS       deprecated
        DESCRIPTION
            "The administrative value that determines what the Aggregator
            does with a frame that is received from an Aggregation Port
            with a Port Conversation ID that is not included in the
```

```
        Collection_Conversation_Mask. The value 'TRUE' indicates that
        such frames are to be discarded, and the value 'FALSE' that
        they are to be forwarded. This variable needs to be set to
        'TRUE', if bidirectional congruity (8.2.1) is required. Its
        value is set to 'TRUE' by default.
        This object has been replaced by
        dot3adAggAdminDiscardWrongConversation2."
    REFERENCE
        "7.3.1.1.37 of IEEE Std 802.1AX-2014"
    ::= { dot3adAggXEntry 24 }

dot3adAggPartnerAdminConvServiceMappingDigest OBJECT-TYPE
    SYNTAX       OCTET STRING (SIZE (16))
    MAX-ACCESS   read-write
    STATUS       deprecated
    DESCRIPTION
        "The value for the digest of the Port Conversation ID-to-Service
        ID assignments of the Partner System, assigned by administrator
        or System policy for use when the Partner's information is
        unknown. Its default value is set to NULL.
        This parameter was eliminated because there was no reason for
        it ever to have a value different from the default value."
    REFERENCE
        "7.3.1.1.39 of IEEE Std 802.1AX-2014"
    DEFVAL  { ''H }
    ::= { dot3adAggXEntry 25 }

dot3adAggAdminDiscardWrongConversation2 OBJECT-TYPE
    SYNTAX       INTEGER {
                    forceTrue(1),
                    forceFalse(2),
                    auto(3)
                 }
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The administrative value that determines whether an Aggregator
        discards a frame that is received from an Aggregation Port
        with a Port Conversation ID that has a value of FALSE in the
        Port_Oper_Conversation_Mask. The value 'forceTrue'
        indicates that such frames are to be discarded, the value
        'forceFalse' indicates that such frames are to be
        forwarded, and the value 'auto' indicates that such frames
        are to be forwarded only when the actor and partner agree
        on the algorithms and mapping tables used to map frames to
        Aggregation Ports. Its value is set to 'auto' by default."
    REFERENCE
        "7.3.1.1.35"
    DEFVAL { auto }
    ::= { dot3adAggXEntry 26 }

dot3adAggOperDiscardWrongConversation OBJECT-TYPE
    SYNTAX       TruthValue
    MAX-ACCESS   read-only
```

```
        STATUS       current
        DESCRIPTION
            "The operational value that determines whether the Aggregator
            discards a frame that is received from an Aggregation Port
            with a Port Conversation ID that has a value of FALSE in the
            Port_Oper_Conversation_Mask."
        REFERENCE
            "7.3.1.1.37"
        ::= { dot3adAggXEntry 27 }

    dot3adAggConvLinkDigest OBJECT-TYPE
        SYNTAX       OCTET STRING (SIZE (16))
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The value for the MD5 digest of aAggAdminConvLinkMap
            (7.3.1.1.34)."
        REFERENCE
            "7.3.1.1.38"
        ::= { dot3adAggXEntry 28 }

    dot3adAggConvServiceDigest OBJECT-TYPE
        SYNTAX       OCTET STRING (SIZE (16))
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The value for the MD5 digest of aAggAdminConvServiceMap
            (7.3.1.1.36)."
        REFERENCE
            "7.3.1.1.39"
        ::= { dot3adAggXEntry 29 }

    dot3adAggPartnerPortAlgorithm OBJECT-TYPE
        SYNTAX       OCTET STRING (SIZE (4))
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "This object identifies the algorithm used by the Partner
            Aggregator to assign frames to a Port Conversation ID.
            A SEQUENCE OF OCTETS consisting of a 3-octet OUI or CID
            and one following octet."
        REFERENCE
            "7.3.1.1.40"
        ::= { dot3adAggXEntry 30 }

    dot3adAggPartnerConvLinkDigest OBJECT-TYPE
        SYNTAX       OCTET STRING (SIZE (16))
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The value for the MD5 digest of the aAggAdminConvLinkMap
            (7.3.1.1.34) in use by the LACP Partner."
        REFERENCE
            "7.3.1.1.41"
```

```
    ::= { dot3adAggXEntry 31 }

dot3adAggPartnerConvServiceDigest OBJECT-TYPE
    SYNTAX       OCTET STRING (SIZE (16))
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The value for the MD5 digest of the aAggAdminConvServiceMap
        (7.3.1.1.36) in use by the LACP Partner."
    REFERENCE
        "7.3.1.1.42"
    ::= { dot3adAggXEntry 32 }

dot3adAggActiveLinks OBJECT-TYPE
    SYNTAX       OCTET STRING
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Each two octets of the octet string represent the operational
        Link_Number of an Aggregation Port that is currently active
        on the Aggregator. An octet string of size zero indicates that
        there are no links currently active on the Aggregator."
    REFERENCE
        "7.3.1.1.43"
    ::= { dot3adAggXEntry 33 }

-- ------------------------------------------------------------------
-- The Aggregation Conversation Admin Link Table
-- ------------------------------------------------------------------

dot3adAggConversationAdminLinkTable OBJECT-TYPE
    SYNTAX       SEQUENCE OF Dot3adAggConversationAdminLinkEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "There are 4096 entries in the table, indexed by
        Port Conversation ID.
        Each contains administrative values of the link selection
        preference list for the referenced Port Conversation ID. This
        selection preference list is a sequence of Link-Numbers for
        each Port Conversation ID, in the order of preference, highest
        to lowest, for the corresponding link to carry that Port
        Conversation ID. A 16-bit zero value is used to indicate
        that no link is assigned to carry the associated Port
        Conversation ID."
    REFERENCE
        "7.3.1.1.34"
    ::= { dot3adAgg 4 }

dot3adAggConversationAdminLinkEntry OBJECT-TYPE
    SYNTAX       Dot3adAggConversationAdminLinkEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
```

```
            "An entry contains administrative values of the link selection
            preference list for the referenced Port Conversation ID. This
            selection preference list is a sequence of Link-Numbers for
            each Port Conversation ID, in the order of preference, highest
            to lowest, for the corresponding link to carry that Port
            Conversation ID. A 16 bit zero value is used to indicate
            that no link is assigned to carry the associated Port
            Conversation ID."
        REFERENCE
            "7.3.1.1.34"
        INDEX { dot3adAggConversationAdminLinkId, dot3adAggIndex}
        ::= { dot3adAggConversationAdminLinkTable 1 }


    Dot3adAggConversationAdminLinkEntry ::=
        SEQUENCE {
            dot3adAggConversationAdminLinkId
                Integer32,
            dot3adAggConversationAdminLinkList
                OCTET STRING
        }

    dot3adAggConversationAdminLinkId OBJECT-TYPE
        SYNTAX      Integer32 (0..4095)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "An identifier for Port Conversation."
        ::= { dot3adAggConversationAdminLinkEntry 1 }


    dot3adAggConversationAdminLinkList OBJECT-TYPE
        SYNTAX      OCTET STRING
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "Each two octets of the octet string represent the agreed
            Link_Number that is assigned to an Aggregation Port
            (7.3.2.1.27). The list is in the order of preference,
            highest to lowest, for corresponding preferred link to
            carry that Port Conversation ID."
        REFERENCE
            "7.3.1.1.34"
        ::= { dot3adAggConversationAdminLinkEntry 2 }

    -- ---------------------------------------------------------------
    -- The Aggregation Admin Service Conversation Map Table
    -- ---------------------------------------------------------------

    dot3adAggAdminServiceConversationMapTable OBJECT-TYPE
        SYNTAX       SEQUENCE OF Dot3adAggAdminServiceConversationMapEntry
        MAX-ACCESS   not-accessible
        STATUS       current
        DESCRIPTION
```

```
        "There are 4096 entries in the table, indexed by Port
        Conversation ID. Each contains, in general, a set of
        32-bit values interpreted as Service Identifiers.
        Service IDs not contained in the map are not
        mapped to any Port Conversation ID and will be discarded."
    REFERENCE
        "7.3.1.1.36"
    ::= { dot3adAgg 5 }

dot3adAggAdminServiceConversationMapEntry OBJECT-TYPE
    SYNTAX Dot3adAggAdminServiceConversationMapEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "An entry contains, in general, a set of Service IDs,
        unique within the array."
    REFERENCE
        "7.3.1.1.36"
    INDEX { dot3adAggAdminServiceConversationMapId, dot3adAggIndex }
    ::= { dot3adAggAdminServiceConversationMapTable 1 }



Dot3adAggAdminServiceConversationMapEntry ::=
    SEQUENCE {
        dot3adAggAdminServiceConversationMapId
            Integer32,
        dot3adAggAdminServiceConversationServiceIDList
            ServiceIdList
    }

dot3adAggAdminServiceConversationMapId OBJECT-TYPE
    SYNTAX      Integer32 (0..4095)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Port Conversation ID used to index Conversation Map
        entries."
    ::= { dot3adAggAdminServiceConversationMapEntry 1}

dot3adAggAdminServiceConversationServiceIDList OBJECT-TYPE
    SYNTAX      ServiceIdList
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "A list contains, in general, a set of Service IDs,
        unique within the array."
    REFERENCE
        "7.3.1.1.36"
    ::= { dot3adAggAdminServiceConversationMapEntry 2}


-- ----------------------------------------------------------
-- The Aggregation Port Table
```

```
-- ---------------------------------------------------------------

dot3adAggPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains Link Aggregation Control
        configuration information about every
        Aggregation Port associated with this device.
        A row appears in this table for each physical port."
    REFERENCE
        "7.3.2"
    ::= { dot3adAggPort 1 }


dot3adAggPortEntry OBJECT-TYPE
    SYNTAX      Dot3adAggPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of Link Aggregation Control configuration
        parameters for each Aggregation Port on this device."
    INDEX       { dot3adAggPortIndex }
    ::= { dot3adAggPortTable 1 }


Dot3adAggPortEntry ::=
    SEQUENCE {
        dot3adAggPortIndex
            InterfaceIndex,
        dot3adAggPortActorSystemPriority
            Integer32,
        dot3adAggPortActorSystemID
            MacAddress,
        dot3adAggPortActorAdminKey
            LacpKey,
        dot3adAggPortActorOperKey
            LacpKey,
        dot3adAggPortPartnerAdminSystemPriority
            Integer32,
        dot3adAggPortPartnerOperSystemPriority
            Integer32,
        dot3adAggPortPartnerAdminSystemID
            MacAddress,
        dot3adAggPortPartnerOperSystemID
            MacAddress,
        dot3adAggPortPartnerAdminKey
            LacpKey,
        dot3adAggPortPartnerOperKey
            LacpKey,
        dot3adAggPortSelectedAggID
            InterfaceIndex,
        dot3adAggPortAttachedAggID
```

```
            InterfaceIndex,
        dot3adAggPortActorPort
            Integer32,
        dot3adAggPortActorPortPriority
            Integer32,
        dot3adAggPortPartnerAdminPort
            Integer32,
        dot3adAggPortPartnerOperPort
            Integer32,
        dot3adAggPortPartnerAdminPortPriority
            Integer32,
        dot3adAggPortPartnerOperPortPriority
            Integer32,
        dot3adAggPortActorAdminState
            LacpState,
        dot3adAggPortActorOperState
            LacpState,
        dot3adAggPortPartnerAdminState
            LacpState,
        dot3adAggPortPartnerOperState
            LacpState,
        dot3adAggPortAggregateOrIndividual
            TruthValue
    }


dot3adAggPortIndex OBJECT-TYPE
    SYNTAX       InterfaceIndex
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The unique identifier allocated to this Aggregation Port by
        the local System. This attribute identifies an Aggregation
        Port instance among the subordinate managed objects of the
        containing object. This value is read-only. NOTE-The aAggPortID
        is represented in the SMIv2 MIB as an ifIndex-see D.4.1."
    REFERENCE
        "7.3.2.1.1"
    ::= { dot3adAggPortEntry 1 }


dot3adAggPortActorSystemPriority OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A 2-octet read-write value used to define the priority
        value associated with the Actor's System ID."
    REFERENCE
        "7.3.2.1.2"
    ::= { dot3adAggPortEntry 2 }


dot3adAggPortActorSystemID OBJECT-TYPE
```

```
        SYNTAX       MacAddress
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A 6-octet read-only MAC address value that defines the
            value of the System ID for the System that contains this
            Aggregation Port."
        REFERENCE
            "7.3.2.1.3"
        ::= { dot3adAggPortEntry 3 }


    dot3adAggPortActorAdminKey OBJECT-TYPE
        SYNTAX       LacpKey
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "The current administrative value of the Key for the
            Aggregation Port. This is a 16-bit read-write value.
            The meaning of particular Key values is of local significance."
        REFERENCE
            "7.3.2.1.4"
        ::= { dot3adAggPortEntry 4 }



    dot3adAggPortActorOperKey OBJECT-TYPE
        SYNTAX       LacpKey
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The current operational value of the Key for the
            Aggregation Port. This is a 16-bit read-only value.
            The meaning of particular Key values is of local significance."
        REFERENCE
            "7.3.2.1.5"
        ::= { dot3adAggPortEntry 5 }



    dot3adAggPortPartnerAdminSystemPriority OBJECT-TYPE
        SYNTAX       Integer32 (0..65535)
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "A 2-octet read-write value used to define the administrative
            value of priority associated with the Partner's System ID. The
            assigned value is used, along with the value of
            aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey,
            aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority,
            in order to achieve manually configured aggregation."
        REFERENCE
            "7.3.2.1.6"
        ::= { dot3adAggPortEntry 6 }
```

```
dot3adAggPortPartnerOperSystemPriority OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 2-octet read-only value indicating the operational value
        of priority associated with the Partner's System ID. The
        value of this attribute contains the manually configured
        value carried in aAggPortPartnerAdminSystemPriority if there
        is no protocol Partner."
    REFERENCE
        "7.3.2.1.7"
    ::= { dot3adAggPortEntry 7 }


dot3adAggPortPartnerAdminSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A 6-octet read-write MACAddress value representing
        the administrative value of the Aggregation Port's protocol
        Partner's System ID. The assigned value is used, along with
        the value of aAggPortPartnerAdminSystemPriority,
        aAggPortPartnerAdminKey, aAggPortPartnerAdminPort,
        and aAggPortPartnerAdminPortPriority, in order to
        achieve manually configured aggregation."
    REFERENCE
        "7.3.2.1.8"
    ::= { dot3adAggPortEntry 8 }


dot3adAggPortPartnerOperSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 6-octet read-only MACAddress value representing
        the current value of the Aggregation Port's protocol Partner's
        System ID. A value of zero indicates that there is no known
        protocol Partner. The value of this attribute contains the
        manually configured value carried in
        aAggPortPartnerAdminSystemID if there is no protocol Partner."
    REFERENCE
        "7.3.2.1.9"
    ::= { dot3adAggPortEntry 9 }


dot3adAggPortPartnerAdminKey OBJECT-TYPE
    SYNTAX       LacpKey
```

```
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "The current administrative value of the Key for the
            protocol Partner. This is a 16-bit read-write value.
            The assigned value is used, along with the value of
            aAggPortPartnerAdminSystemPriority,
            aAggPortPartnerAdminSystemID, aAggPortPartnerAdminPort,
            and aAggPortPartnerAdminPortPriority, in order to achieve
            manually configured aggregation."
        REFERENCE
            "7.3.2.1.10"
        ::= { dot3adAggPortEntry 10 }


    dot3adAggPortPartnerOperKey OBJECT-TYPE
        SYNTAX       LacpKey
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The current operational value of the Key for the
            protocol Partner. The value of this attribute contains
            the manually configured value carried in
            aAggPortPartnerAdminKey if there is no protocol Partner.
            This is a 16-bit read-only value."
        REFERENCE
            "7.3.2.1.11"
        ::= { dot3adAggPortEntry 11 }


    dot3adAggPortSelectedAggID OBJECT-TYPE
        SYNTAX       InterfaceIndex
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The identifier value of the Aggregator that this Aggregation
            Port has currently selected. Zero indicates that the
            Aggregation Port has not selected an Aggregator, either because
            it is in the process of detaching from an Aggregator or because
            there is no suitable Aggregator available for it to select.
            This value is read-only."
        REFERENCE
            "7.3.2.1.12"
        ::= { dot3adAggPortEntry 12 }


    dot3adAggPortAttachedAggID OBJECT-TYPE
        SYNTAX       InterfaceIndex
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The identifier value of the Aggregator that this Aggregation
            Port is currently attached to. Zero indicates that the
            Aggregation Port is not currently attached to an Aggregator.
```

```
        This value is read-only."
    REFERENCE
        "7.3.2.1.13"
    ::= { dot3adAggPortEntry 13 }


dot3adAggPortActorPort OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The port number locally assigned to the Aggregation Port.
        The port number is communicated in LACPDUs as the
        Actor_Port. This value is read-only."
    REFERENCE
        "7.3.2.1.14"
    ::= { dot3adAggPortEntry 14 }


dot3adAggPortActorPortPriority OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The priority value assigned to this Aggregation Port.
        This 16-bit value is read-write."
    REFERENCE
        "7.3.2.1.15"
    ::= { dot3adAggPortEntry 15 }


dot3adAggPortPartnerAdminPort OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The current administrative value of the port number
        for the protocol Partner. This is a 16-bit read-write value.
        The assigned value is used, along with the value of
        aAggPortPartnerAdminSystemPriority,
        aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey,
        and aAggPortPartnerAdminPortPriority,
        in order to achieve manually configured aggregation."
    REFERENCE
        "7.3.2.1.16"
    ::= { dot3adAggPortEntry 16 }


dot3adAggPortPartnerOperPort OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The operational port number assigned to this Aggregation
```

```
        Port by the Aggregation Port's protocol Partner. The value
        of this attribute contains the manually configured value
        carried in aAggPortPartnerAdminPort if there is no protocol
        Partner. This 16-bit value is read-only."
    REFERENCE
        "7.3.2.1.17"
    ::= { dot3adAggPortEntry 17 }


dot3adAggPortPartnerAdminPortPriority OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The current administrative value of the port priority
        for the protocol Partner. This is a 16-bit read-write value.
        The assigned value is used, along with the value of
        aAggPortPartnerAdminSystemPriority,
        aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey, and
        aAggPortPartnerAdminPort, in order to achieve manually
        configured aggregation."
    REFERENCE
        "7.3.2.1.18"
    ::= { dot3adAggPortEntry 18 }


dot3adAggPortPartnerOperPortPriority OBJECT-TYPE
    SYNTAX       Integer32 (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The priority value assigned to this Aggregation Port
        by the Partner. The value of this attribute contains the
        manually configured value carried in
        aAggPortPartnerAdminPortPriority if there is no
        protocol Partner. This 16-bit value is read-only."
    REFERENCE
        "7.3.2.1.19"
    ::= { dot3adAggPortEntry 19 }


dot3adAggPortActorAdminState OBJECT-TYPE
    SYNTAX       LacpState
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A string of 8 bits, corresponding to the administrative
        values of the Actor state (Figure 6-9) as transmitted by the
        Actor in LACPDUs. The first bit corresponds to bit 0 of the
        Actor_State (LACP_Activity), the second bit corresponds to
        bit 1 (Short_Timeout), the third bit corresponds to bit 2
        (Aggregation), the fourth bit corresponds to bit 3
        (Synchronization), the fifth bit corresponds to bit 4
        (Collecting), the sixth bit corresponds to bit 5
```