

802.3ae™

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

**Part 3: Carrier Sense Multiple Access with
Collision Detection (CSMA/CD) Access Method
and Physical Layer Specifications**

**Amendment: Media Access Control (MAC)
Parameters, Physical Layers, and Management
Parameters for 10 Gb/s Operation**

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee



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

30 August 2002

Print: SH94996
PDF: SS94996

SUPERSEDED*

Recognized as an
American National Standard (ANSI)

IEEE Std 802.3ae™-2002
(Amendment to IEEE Std 802.3™-2002)

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

Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications

Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Param- eters for 10 Gb/s Operation

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

Approved 17 December 2002
American National Standards Institute

Approved 13 June 2002
IEEE-SA Standards Board

Abstract: Support to extend the IEEE 802.3 protocol and MAC specification to an operating speed of 10 Gb/s. Several Physical Coding Sublayers known as 10GBASE-X, 10GBASE-R, and 10GBASE-W are specified, as well as significant additional supporting material for a 10 Gigabit Media Independent Interface (XGMII), a 10 Gigabit Attachment Unit Interface (XAUI), a 10 Gigabit Sixteen-Bit Interface (XSBI) and management.

Keywords: 10 Gigabit Ethernet, fiber, LAN PHY, WAN PHY, XAUI, XGMII, XSBI

"Superseded" standards are standards that are no longer useful or contain significant obsolete or erroneous information. All amendments or corrigenda associated with this standard have also been superseded.

WARNING: You are cautioned TO USE AT YOUR OWN RISK, as the technology contained herein may no longer be accurate or may have been replaced.

Information on superseded standards is available in the IEEE-SA Standards Board Operations Manual
<<http://standards.ieee.org/guides/opman/index.html>>.

SUPERSEDED

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

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

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

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

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

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

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

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

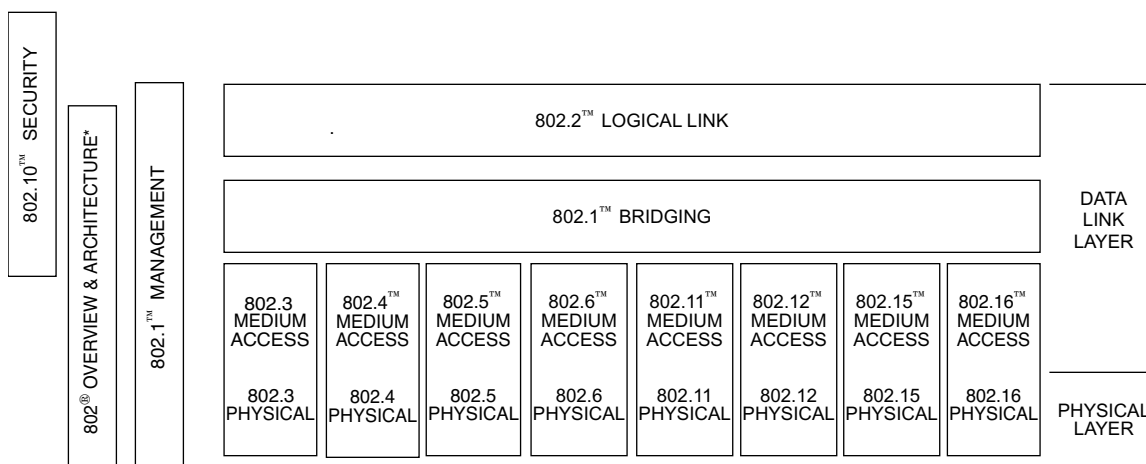
Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

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

SUPERSEDED

Introduction to IEEE Std 802.3ae-2002

This standard is part of a family of standards for local and metropolitan area networks. The relationship between the standard and other members of the family is shown below. (The numbers in the figure refer to IEEE standard numbers.¹)



* Formerly IEEE Std 802.1A™

This family of standards deals with the Physical and Data Link layers as defined by the International Organization for Standardization (ISO) Open Systems Interconnection (OSI) Basic Reference Model (ISO/IEC 7498-1: 1994). The access standards define seven types of medium access technologies and associated physical media, each appropriate for particular applications or system objectives. Other types are under investigation.

The standards defining the technologies noted above are as follows:

- IEEE Std 802 *Overview and Architecture.* This standard provides an overview to the family of IEEE 802 Standards.
- IEEE Std 802.1B™ and 802.1k™ [ISO/IEC 15802-2] *LAN/MAN Management.* Defines an OSI management-compatible architecture, and services and protocol elements for use in a LAN/MAN environment for performing remote management.
- IEEE Std 802.1D™ *Media Access Control (MAC) Bridges.* Specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary.
- IEEE Std 802.1E™ [ISO/IEC 15802-4] *System Load Protocol.* Specifies a set of services and protocol for those aspects of management concerned with the loading of systems on IEEE 802 LANs.
- IEEE Std 802.1F™ *Common Definitions and Procedures for IEEE 802 Management Information.*
- IEEE Std 802.1G™ [ISO/IEC 15802-5] *Remote Media Access Control (MAC) Bridging.* Specifies extensions for the interconnection, using non-LAN communication technologies, of geographically separated IEEE 802 LANs below the level of the logical link control protocol.
- IEEE Std 802.1H™ [ISO/IEC TR 11802-5] *Media Access Control (MAC) Bridging of Ethernet V2.0 in Local Area Networks.*

¹The IEEE standards referred to in the above figure and list are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

SUPERSEDED

- IEEE Std 802.2 [ISO/IEC 8802-2] *Logical Link Control.*
- IEEE Std 802.3 *CSMA/CD Access Method and Physical Layer Specifications.*
- IEEE Std 802.4 [ISO/IEC 8802-4] *Token Passing Bus Access Method and Physical Layer Specifications.*
- IEEE Std 802.5 [ISO/IEC 8802-5] *Token Ring Access Method and Physical Layer Specifications.*
- IEEE Std 802.6 [ISO/IEC 8802-6] *Distributed Queue Dual Bus Access Method and Physical Layer Specifications.*
- IEEE Std 802.10 *Interoperable LAN/MAN Security.*
- IEEE Std 802.11 [ISO/IEC DIS 8802-11] *Wireless LAN Medium Access Control (MAC) and Physical Layer Specifications.*
- IEEE Std 802.12 [ISO/IEC 8802-12] *Demand Priority Access Method, Physical Layer and Repeater Specifications.*
- IEEE Std 802.15 *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for: Wireless Personal Area Networks.*
- IEEE Std 802.16 *Standard Air Interface for Fixed Broadband Wireless Access Systems.*

In addition to the family of standards, the following is a recommended practice for a common Physical Layer technology:

- IEEE Std 802.7™ *IEEE Recommended Practice for Broadband Local Area Networks.*

Conformance test methodology

An additional standard, IEEE Std 1802.3™ provides conformance test information for 10BASE-T.

IEEE Std 802.3ae-2002

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated to this standard within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material. Details on the contents of this standard are provided on the following pages.

SUPERSEDED

Participants

The following is a list of chairs and editors at the time the IEEE 802.3 Working Group balloted this standard:

Geoffrey O. Thompson, *Chair*

David J. Law, *Vice Chair*

Robert M. Grow, *Secretary*

R. Jonathan Thatcher, *Chair, 802.3ae Task Force*

Stephen Haddock, *Vice Chair, 802.3ae Task Force*

Bradley J. Booth, *Editor, 802.3ae Task*

Lacreshia Laningham, *Assistant Editor, 802.3ae Task Force*

Benjamin Brown, *Logic Track Chair, 802.3ae Task Force*

Walter Thirion, *Optical Track Chair, 802.3ae Task Force*

Shimon Muller, *Clauses 1, 2, 4, 6, 22, 31, 35, and Annex 31B*

David J. Law, *Clause 30 and Annexes 30 A and 30B*

Bradley J. Booth, *Clause 44 and Annex 44A*

Edward Turner, *Clause 45 and Annex 45A*

Robert M. Grow, *Clause 46*

Dawson Kesling, *Clause 47*

Rich Taborek, *Clause 48 and Annexes 48A and 48B*

Eric R. Lynskey, *Clause 48 and Annexes 48A and 48B*

Pat Thaler, *Clause 49*

Tom Alexander, *Clause 50 and Annex 50A*

Justin Chang, *Clause 51*

David Kabal, *Clause 52*

Eric Grann, *Clause 53*

The following is a list of voters at the time the IEEE 802.3 Working Group balloted this standard:

Oscar Agazzi
Don Alderrou
Thomas Alexander
Khaled Amer
Keith Amundsen
Ole Christian Andersen
Arlan J. Anderson
Eric Anderson
Ralph Andersson
Jack Andresen
Mehran Ataeae
Phil Auld
Kameran Azadet
Joseph N. Babanezhad
Gerard E. Bachand
Andy Baldman
Howard Baumer
Denis Beaudoin
Mike Bennett
Sidney Berglund
John L. Bestel
Vipul Bhatt
Michel Bohbot
Bradley J. Booth
Paul Bottorff
Gary Bourque
Kirk Bovill
Andrew Brierley-Green
Rhett Brikovskis
Benjamin Brown
Dave Brown
Kevin Brown
Steve F. Buck
Lisa Buckman
James Burgess
Scott Burton
Robert Busse
Roy Bynum
Richard Cam
Bob Campbell
Steve Carlson
Edward G. Chang
Edward S. Chang

Justin Chang
Xiaopeng Chen
Zinan (Nan) Chen
Hon Wah Chin
Kuen Chow
George Claseman
Terry Cobb
Doug Coleman
Régis Colla
Herb Congdon
Edward Cornejo
Richard Cross
Brian Cruikshank
Chris Cullin
David Cunningham
Robert Dahlgren
Kevin Daines
John Dallesasse
Rupert S. Dance
Yair Darshan
Peter Dartnell
Piers Dawe
Michael de la Garrigue
Tom Debiec
Joel Dedrick
Chris Di Minico
Wael Diab
Thomas Dineen
Hamish Dobson
David W. Dolfi
Mark Donhowe
Dan Dove
Daniel S. Draper
Mike Dudek
Richard Dugan
Marc R. Dupuis
Clay Eddings
George Eisler
Martin Elhøj
John F. Ewen
Mark Feuerstraeter
Jens Fiedler
Norival Figueira

Farzin Firoozmand
Alan Flatman
Howard Frazier
Ladd Freitag
Scott Fritz
Krister Frojdh
Darrell Furlong
Justin Gaither
John George
Pat Gilliland
Joel Goergen
Moty Goldis
Rich Graham
Eric Grann
C. Thomas Gray
Jonathan E. Greenlaw
Robert M. Grow
Michael Hackert
Stephen Haddock
Sharam Hakimi
Farid Hamidy
Johannes Hansen
Del Hanson
Marwan Hassoun
Tom Hatley
Adam Healey
Ronen Heldman
Itzik Hendel
Ken Herrity
James H. Hesson
Henry Hinrichs
David Hinzal
Jay Hoge
David W. Hyer
Haruhiko Ichino
Osamu Ishida
Steve Jackson
Mike Jacobson
Eric Jang
Woo-Hyuk Jang
Ernie Jensen
Jack L. Jewell
Wenbin Jiang

Clarence Joh
Thomas K. Jørgensen
David Kabal
Shinkyo Kaku
Mohan Kalkunte
Puru Kamat
Roger Karam
Jaime Kardontchik
Toyoyuki Kato
Kevin Kayser
N. Patrick Kelly
Dawson Kesling
Dae Young Kim
Yongbum Kim
David E. Kohl
Paul Kolesar
David Kooistra
Lars Paul Krolner
Pankaj Kumar
Hans Lackner
William Lane
Daun Langston
Donald C. Larson
Ryan Latchman
David J. Law
Changoo Lee
Hyeong Ho Lee
Wesley Lee
Amir Lehr
Brian E. Lemoff
Lisa Leo
Robert H. Leonowich
Michael Lerer
Avinoam Levy
Van Lewing
Bob Love
Rick Loveless
Fred A. Lucas
Meilissa R. Lum
Jeffrey Lynch
Eric R. Lynskey
Henning Lysdal
Brian MacLeod

SUPERSEDED

David W. Martin
Thomas Mathey
Bob Mayer
Philip L. McCarron
Michael S. McCormack
Gary McCoy
Joseph Micallef
Martin R. Milbury
Larry D. Miller
Reza Moattar
Fred Mohamadi
Dirk S. Mohl
Cindy Montstream
Paul B. Moore
Robert Moore
Robert Muir
Shimon Muller
Gerard Nadeau
Ken Naganuma
Hari Naidu
Karl Nakamura
Nersi Nazari
Kristian Nelson
Paul Nikolich
Glenn Nishida
Michael Nootbaar
Bob Noseworthy
Michael O'Toole
Satoshi Obara
Stephen Oh
Peter Ohlén
George Oughton

Robert R. Pace
Tom Palkert
Don Pannell
Keshab K. Parhi
Elwood T. Parsons
Joel Paslaski
Bhavesh Patel
Dipak M. Patel
Alex Pavlovsky
John Payne
Petar Pepeljugoski
Abhijit Phanse
Armin Pitzer
Timothy R. Plunkett
Peter Pondillo
Jeff Porter
Bernd Prediger
William Quackenbush
John Quirk
Rick Rabinovich
Sailesh K. Rao
Jennifer G. Rasimas
Dan Rausch
Peter Rautenberg
Maurice Reintjes
Lawrence Rennie
Dave Richkas
Ramez Rizk
Gary Robinson
Stuart Robinson
Shawn Rogers
Tume Römer
Floyd Ross
Larry Rubin

Anthony Sanders
Raj Savara
Thomas Schramm
Benjamin Schultz
Klaus Schulz
Peter Schwartz
Lee Sendelbach
Koichiro Seto
Vadim Shain
Tim Simmons
David Sorensen
Nick Stapleton
Claus Stetter
Donald S. Stewart
Mario Stoltz
Tetsuyuki Suzuki
Hiroshi Suzuki
Daniel Svensson
Steve Swanson
Tad Szostak
Rich Taborek
Bharat Tailor
Akio Tajima
Mike Tate
James M. Tavacoli
Pat Thaler
R. Jonathan Thatcher
Walter Thirion
Geoffrey Thompson
Douglas Thomson
Bruce Tolley
Paul Torgerson
Luis Torres
Thomas E. Truman

Edward Turner
Bor-long Twu
Sterling A. Vaden
Schelto van Doorn
Erik van Oosten
Gérard Vergnaud
Nader Vjeh
Ron Villozny
David Vogel
Martin Wagner
Rick Walker
Peter Wang
Tim Warland
Jeff Warren
Ted Washburn
Yuji Watanabe
Fred Weniger
Willem Wery
Tony Whitlow
Bill Wiedemann
Robert S. Williamson
Mike Witkowski
John Wolcott
Jonghwa Won
King Won
Edward Wong
Leo Wong
Stefan M. Wurster
Jason Yorks
Takashi Yoshikawa
Leonard Young
Nariman Yousefi
Hank Zannini

SUPERSEDED

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

Don Alderrou	Osamu Ishida	Juergen Rahn
Thomas Alexander	Stephen Jackson	Narayanan Raman
Jacob Ben Ary	Brent E Jaffa	Maurice Reintjes
Mike Bennett	Raj Jain	Stephan Reis
Rahul Bhushan	David Kabal	Lawrence Rennie
Bradley J. Booth	Paul Kolesar	Calvin Roberts
Paul Bottorff	William Lane	David Rockwell
Benjamin Brown	David J. Law	Floyd Ross
Roy Bynum	Quang Le	Thomas Ruf
Simin Cai	Steven Levey	Raj Savara
Robert Carlisle	Tom Lindsay	Steven Selee
Jim Carlo	Gregory Luri	Leo Sintonen
Linda Cheng	Jeffrey Lynch	Charles Spurgeon
Keith Chow	James Markevitch	Steven Swanson
John Dallesasse	Peter Martini	Richard Taborek, Sr.
Piers Dawe	Kyle Maus	Pat Thaler
Thomas Dineen	Steve Methley	R. Jonathan Thatcher
Christos Douligeris	Hiroshi Miyano	Geoffrey O. Thompson
Michael (Mike) Dudek	Robert Mortonson	Jerry Thrasher
John Fendrich	Shimon Muller	Edward Turner
Robert Finch	Hari Naidu	Scott Valcourt
Justin Gaither	M. Ravindranath Nayak	Peter Van Laanen
Geoffrey Garner	Paul Nikolich	Ramkrishna Vepa
Patrick Gonia	Ellis Nolley	Tim Warland
Jonathan Greenlaw	Bob O'Hara	Jay Warrior
Robert M. Grow	Satoshi Obara	Fred Weniger
Chris Guy	Peter Ohlen	King Won
Stephen Haddock	Roger Pandanda	Stefan Wurster
Sharam Hakimi	Petar Pepeljugoski	Leonard Young
John Hawkins	Jeff Porter	Oren Yuen
Robert Hughes	Vikram Punj	Robert Zona

When the IEEE-SA Standards Board approved this standard on 13 June 2002, it had the following membership:

James T. Carlo, *Chair*
James H. Gurney, *Vice Chair*
Judith Gorman, *Secretary*

Sid Bennett	Toshio Fukuda	Nader Mehravari
H. Stephen Berger	Arnold M. Greenspan	Daleep C. Mohla
Clyde R. Camp	Raymond Hapeman	William J. Moylan
Richard DeBlasio	Donald M. Heirman	Malcolm V. Thaden
Harold E. Epstein	Richard H. Hulett	Geoffrey O. Thompson
Julian Forster*	Lowell G. Johnson	Howard L. Wolfman
Howard M. Frazier	Joseph L. Koepfinger*	Don Wright
	Peter H. Lips	

*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

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

Catherine K.N. Berger
IEEE Standards Project Editor

Contents

REVISIONS TO IEEE Std 802.3-2002

1.	(Changes to) Introduction	2
1.1	(Changes to) Overview	2
1.3	(Changes to) Normative references	4
1.4	(Changes to) Definitions	5
1.5	(Changes to) Abbreviations	7
2.	(Changes to) Media Access Control (MAC) service specification	9
2.1	(Changes to) Scope and field of application	9
4.	(Changes to) Media Access Control	13
6.	(Changes to) Physical Signaling (PLS) service specifications	41
6.1	(Changes to) Scope and field of application	41
22.	(Changes to) Reconciliation Sublayer (RS) and Media Independent Interface (MII)	43
22.1	(Changes to) Overview	43
30.	(Changes to) 10 Mb/s, 100 Mb/s, 1000 Mb/s, and 10 Gb/s MAC Control, and Link Aggregation Management	45
30.1	(Changes to) Overview	45
30.5	(Changes to) Layer management for 10, 100, and 1000 Mb/s medium attachment units (MAUs)	53
30.8	(Changes to) Management for WAN Interface Sublayer (WIS)	57
31.	(Changes to) MAC Control	65
31.3	(Changes to) Support by interlayer interfaces	65
35.	(Changes to) Reconciliation Sublayer (RS) and Gigabit Media Independent Interface (GMII)	67
35.1	(Changes to) Overview	67
ANNEXES		
(Changes to) Annex 30A (normative) GDMO specification for IEEE 802.3 managed object classes		69
(Changes to) Annex 30B (normative) GDMO and ASN.1 definitions for management		141
(Changes to) Annex 31B (normative) MAC Control PAUSE operation		149

44.	Introduction to 10 Gb/s baseband network	153
44.1	Overview	153
44.2	State diagrams	156
44.3	Delay constraints	156
44.4	Protocol Implementation Conformance Statement (PICS) proforma	156
44.5	Relation of 10 Gigabit Ethernet to other standards	158
45.	Management Data Input/Output (MDIO) Interface	161
45.1	Overview	161
45.2	MDIO Interface Registers	162
45.3	Management frame structure	225
45.4	Electrical interface	227
45.5	Protocol Implementation Conformance Statement (PICS) proforma for Clause 45, MDIO interface	229
46.	Reconciliation Sublayer (RS) and 10 Gigabit Media Independent Interface (XGMII)	247
46.1	Overview	247
46.2	XGMII data stream	252
46.3	XGMII functional specifications	254
46.4	XGMII electrical characteristics	263
46.5	Protocol Implementation Conformance Statement (PICS) proforma for Clause 46, Reconciliation Sublayer (RS) and 10 Gigabit Media Independent Interface (XGMII)	265
47.	XGMII Extender Sublayer (XGXS) and 10 Gigabit Attachment Unit Interface (XAUI)	271
47.1	Overview	271
47.2	Functional specifications	272
47.3	XAUI Electrical characteristics	273
47.4	Electrical measurement requirements	279
47.5	Environmental specifications	280
47.6	Protocol Implementation Conformance Statement (PICS) proforma for Clause 47, XGMII Extender (XGMII) and 10 Gigabit Attachment Unit Interface (XAUI)	281
48.	Physical Coding Sublayer (PCS) and Physical Medium Attachment (PMA) sublayer, type 10GBASE-X285	
48.1	Overview	285
48.2	Physical Coding Sublayer (PCS)	289
48.3	Physical Medium Attachment (PMA) sublayer	310
48.4	Compatibility considerations	313
48.5	Delay constraints	313
48.6	Environmental specifications	313
48.7	Protocol Implementation Conformance Statement (PICS) proforma for Physical Coding Sublayer (PCS) and Physical Medium Attachment (PMA) sublayer, type 10GBASE-X	314
49.	Physical Coding Sublayer (PCS) for 64B/66B, type 10GBASE-R	317
49.1	Overview	317
49.2	Physical Coding Sublayer (PCS)	321
49.3	Protocol Implementation Conformance Statement (PICS) proforma for Clause 49, Physical Coding Sublayer (PCS) type 10GBASE-R	341

50.	WAN Interface Sublayer (WIS), type 10GBASE-W	347
50.1	Overview	347
50.2	WIS Service Interface	350
50.3	Functions within the WIS	353
50.4	Synchronization state diagram	371
50.5	Environmental specifications	375
50.6	Protocol Implementation Conformance Statement (PICS) proforma for Clause 50, WAN Interface Sublayer (WIS), type 10GBASE-W	377
51.	Physical Medium Attachment (PMA) sublayer, type Serial	383
51.1	Overview	383
51.2	PMA Service Interface	384
51.3	Functions within the PMA	385
51.4	Sixteen-Bit Interface (XSBI)	386
51.5	General electrical characteristics of the XSBI	390
51.6	XSBI transmit interface electrical characteristics	392
51.7	XSBI receive interface electrical characteristics	394
51.8	PMA Loopback mode (optional)	396
51.9	Environmental specifications	397
51.10	Protocol Implementation Conformance Statement (PICS) proforma for Clause 51, Physical Medium Attachment (PMA) sublayer, type Serial	398
52.	Physical Medium Dependent (PMD) sublayer and baseband medium, type 10GBASE-S (Short Wavelength Serial), 10GBASE-L (Long Wavelength Serial), and 10GBASE-E (Extra Long Wavelength Serial)	401
52.1	Overview	401
52.2	Delay constraints	404
52.3	PMD MDIO function mapping	404
52.4	PMD functional specifications	405
52.5	PMD to MDI optical specifications for 10GBASE-S	407
52.6	PMD to MDI optical specifications for 10GBASE-L	409
52.7	PMD to MDI optical specifications for 10GBASE-E	413
52.8	Jitter specifications for 10GBASE-R and 10GBASE-W	415
52.9	Optical measurement requirements	417
52.10	Environmental specifications	432
52.11	Environment	432
52.12	PMD labeling requirements	433
52.13	Fiber optic cabling model	433
52.14	Characteristics of the fiber optic cabling (channel)	434
52.15	Protocol Implementation Conformance Statement (PICS) proforma for Clause 52, Physical Medium Dependent (PMD) sublayer and baseband medium, type 10GBASE-S (Short Wavelength Serial), 10GBASE-L (Long Wavelength Serial), and 10GBASE-E (Extra Long Wavelength Serial)	437
53.	Physical Medium Dependent (PMD) sublayer and baseband medium, type 10GBASE-LX4	445
53.1	Overview	445
53.2	Delay constraints	447
53.3	PMD MDIO function mapping	448
53.4	PMD functional specifications	449
53.5	Wavelength-division multiplexed-lane assignments	452

53.6	Operating ranges for 10GBASE-LX4 PMD	452
53.7	PMD to MDI optical specifications for 10GBASE-LX4.....	453
53.8	Jitter specifications for each lane of the 10GBASE-LX4 PMD	454
53.9	Optical measurement requirements.....	458
53.10	Environmental specifications	470
53.11	Environment.....	471
53.12	PMD labeling requirements	471
53.13	Fiber optic cabling model	472
53.14	Characteristics of the fiber optic cabling (channel)	472
53.15	Protocol Implementation Conformance Statement (PICS) proforma for Clause 53, Physical Medium Dependent (PMD) sublayer and baseband medium, type 10GBASE-LX4	475
Annex 44A (informative) Diagram of Data Flow.....		485
Annex 45A (informative) Clause 45 MDIO electrical interface		493
Annex 48A (normative) Jitter test patterns		497
Annex 48B (informative) Jitter test methods.....		505
Annex 50A (informative) Thresholds for Severely Errored Second calculations		515

SUPERSEDED

SUPERSEDED

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

Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications

Changes to IEEE Std 802.3™-2002

[These changes are part of IEEE Std 802.3-2002.]

EDITORIAL NOTE—This amendment is based on the current edition of IEEE Std 802.3-2002. The editing instructions define how to merge the material contained here into this base document set to form the new comprehensive standard as created by the addition of IEEE Std 802.3ae-2002.

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

1. Introduction

1.1 Overview

Change 1.1 to read as follows:

This is a comprehensive International Standard for Local and Metropolitan Area Networks (LANs and MANs), employing CSMA/CD as the shared media access method and the IEEE 802.3 (Ethernet) protocol and frame format for data communication. This International Standard is intended to encompass several media types and techniques for a variety of signal rates, from 1Mb/s to 1000Mb/s. This edition of the standard provides the necessary specifications for the following families of systems: a 1Mb/s baseband system, 10Mb/s baseband and broadband systems, a 100Mb/s baseband system, and a 1000Mb/s baseband system. The families of systems supported by this standard are shown in Figure 1–1 and listed in 4.4.2. In addition, it specifies a method for linearly incrementing a system's data rate by aggregating multiple physical links of the same speed into one logical link.

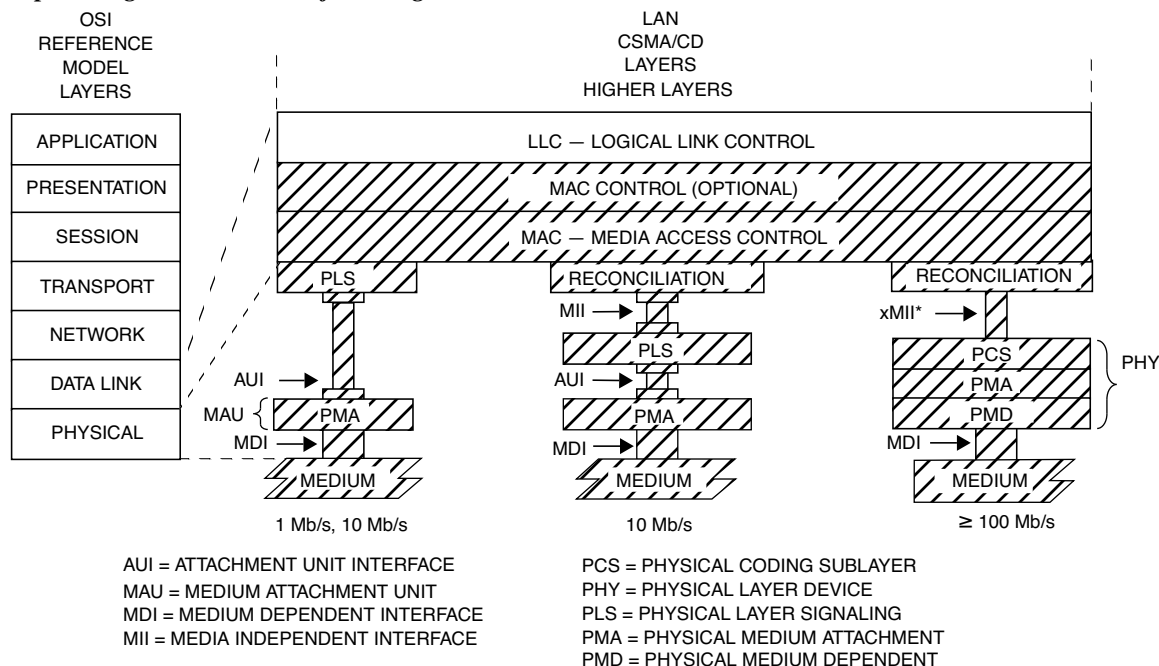
1.1.1.1 Half duplex operation

Change the second paragraph in 1.1.1.1 to read as follows:

Half duplex operation can be used with all certain media types and configurations allowed as defined by this standard. For allowable configurations, see 4.4.2.

1.1.2 Architectural perspectives

Replace Figure 1–1 with the following:



NOTE—In this figure, the xMII is used as a generic term for the Media Independent Interfaces for implementations of 100 Mb/s and above. For example: for 100 Mb/s implementations this interface is called MII; for 1 Gb/s implementations it is called GMII; for 10 Gb/s implementations it is called XGMII; etc.

Figure 1–1 —IEEE 802.3 standard relationship to the ISO/IEC Open Systems Interconnection (OSI) reference model

Change the first sentence in 1.1.2 to read as follows:

There are two important ways to view local-area-network (LAN)-design, corresponding to the following:

1.1.2.2 Compatibility interfaces

Change the first sentence in 1.1.2.2 to read as follows:

~~Five~~The following important compatibility interfaces are defined within what is architecturally the Physical Layer.

Change item d) in 1.1.2.2 to read as follows:

- d) Gigabit Media Independent Interface (GMII). The GMII is designed to connect a 1 Gb/s capable ~~gigabit-capable~~ MAC or repeater unit to a 1 Gb/s ~~gigabit~~ PHY. While conformance with implementation of this interface is not strictly necessary to ensure communication, it is highly recommended, since it allows maximum flexibility in intermixing PHYs and DTEs at 1 Gb/s ~~gigabit~~ speeds. The GMII is intended for use as a chip-to-chip interface. No mechanical connector is specified for use with the GMII. The GMII is optional.

Insert the following lettered items at the end of subclause 1.1.2.2:

- f) 10 Gigabit Media Independent Interface (XGMII). The XGMII is designed to connect a 10 Gb/s capable MAC to a 10 Gb/s PHY. While conformance with implementation of this interface is not strictly necessary to ensure communication, it is highly recommended, since it allows maximum flexibility in intermixing PHYs and DTEs at 10 Gb/s speeds. The XGMII is intended for use as a chip-to-chip interface. No mechanical connector is specified for use with the XGMII. The XGMII is optional.
- g) 10 Gigabit Attachment Unit Interface (XAUI). The XAUI is designed to extend the connection between a 10 Gb/s capable MAC and a 10 Gb/s PHY. While conformance with implementation of this interface is not strictly necessary to ensure communication, it is highly recommended, since it allows maximum flexibility in intermixing PHYs and DTEs at 10 Gb/s speeds. The XAUI is intended for use as a chip-to-chip interface. No mechanical connector is specified for use with the XAUI. The XAUI is optional.
- h) 10 Gigabit Sixteen-Bit Interface (XSBI). The XSBI is provided as a physical instantiation of the PMA service interface for 10GBASE-R and 10GBASE-W PHYs. While conformance with implementation of this interface is not strictly necessary to ensure communication, it is highly recommended, since it provides a convenient partition between the high-frequency circuitry associated with the PMA sublayer and the logic functions associated with the PCS and MAC sublayers. The XSBI is intended for use as a chip-to-chip interface. No mechanical connector is specified for use with the XSBI. The XSBI is optional.

1.1.4 Application areas

Change 1.1.4 to read as follows:

The applications environment for the LAN is intended to be commercial and light industrial. Use of CSMA/CD LANs in home or heavy industrial environments, while not precluded, is not considered within the scope of this standard.

Use of this standard is not restricted to any specific environments or applications.

In the context of this standard, the term “LAN” is used to indicate all networks that utilize the 802.3 (Ethernet) protocol for communication. These may include (but are not limited to) LANs and MANs.

1.2.1 State diagram conventions

Change the fifth paragraph in 1.2.1 to read as follows:

Some devices described in this standard (e.g., repeaters) are allowed to have two or more ports. State diagrams that are capable of describing the operation of devices with an unspecified number of ports, require a qualifier notation that allows testing for conditions at multiple ports. The notation used is a term that includes a description in parentheses of which ports must meet the term for the qualifier to be satisfied (e.g., ANY and ALL). It is also necessary to provide for term-assignment statements that assign a name to a port that satisfies a qualifier. The following convention is used to describe a term-assignment statement that is associated with a transition:

Add the following subclause after 1.2.4:

1.2.5 Hexadecimal notation

Numerical values designated by the 0x prefix indicate a hexadecimal interpretation of the corresponding number. For example: 0x0F represents an 8-bit hexadecimal value of the decimal number 15; 0x00000000 represents a 32-bit hexadecimal value of the decimal number 0; etc.

1.3 Normative references

Insert the following references in alphabetic order into the reference list in 1.3:

ANSI T1.105-1995, Telecommunications—Synchronous Optical Network (SONET)—Basic Description including Multiplex Structure, Rates and Formats.

ANSI T1.269-2000, Information Interchange—Structure and Representation of Trace Message formats for the North American Telecommunications System.

ANSI T1.416-1999, Telecommunications—Network to Customer Installation Interfaces—Synchronous Optical Network (SONET) Physical Layer Specification: Common Criteria.

ANSI X3.230-1994 (FC-PH), Information Technology—Fibre Channel—Physical and Signaling Interface.

ANSI/TIA/EIA-455-175A-92; Chromatic Dispersion Measurement of Single-Mode Optical Fibers by the Differential Phase-Shift Method.

ANSI/TIA/EIA-455-203-2001; Launched Power Distribution Measurement Procedure for Graded-Index Multimode Transmitters.

ANSI/TIA/EIA-455-204-2000; Measurement of Bandwidth on Multimode Fiber.

ANSI/TIA/EIA-526-4A-1997 (OFSTP-4A), Optical Eye Pattern Measurement Procedure.

ANSI/TIA/EIA-568-B.3-2000; Optical Fiber Cabling Components Standard.

EIA/JEDEC Standard EIA/JESD8-6, High Speed Transceiver Logic (HSTL), August 1995.¹

IEC 60793-1-41: 2001, Optical fibres—Part 1-41: Measurement methods and test procedures—Bandwidth.

IEC 60825-1: 2001, Edition 1.2, Consolidated Edition; Safety of Laser Products—Part 1: Equipment classification, requirements and user's guide.

IEC 61753-1-1: 2000, Fibre optic interconnecting devices and passive components performance standard - Part 1-1: General and guidance—Interconnecting devices (connectors).

IEC 61753-021-2 (2002-07), Fibre optic passive component performance standard—Part 021-2: Fibre optic connectors terminated on single-mode fibre to category C Controlled environment.

IEC 61753-022-2, Performance standard—Part 022-2: Fibre optic connectors terminated on multimode fibre for Category C—Controlled environment.²

INCITS-TR-25:1999—Fibre Channel Methodologies for Jitter Specification.³

ITU-T Recommendation G.652, 2000—Characteristics of a single-mode optical fibre cable.

ITU-T Recommendation G.691, 2000—Optical interfaces for single-channel STM-64, STM-256 and other SDH systems with optical amplifiers.

ITU-T Recommendation G.957, 1999—Optical interfaces for equipments and systems relating to the synchronous digital hierarchy (SDH).

ITU-T Recommendation O.150, 1996—General requirements for instrumentation for performance measurements on digital transmission equipment.

ITU-T Recommendation O.172, 1999—Jitter and wander measuring equipment for digital systems which are based on the synchronous digital hierarchy (SDH).

TIA-492AAAC-2002; Detail Specification for 850-nm Laser-Optimized, 50-um core diameter/125-um cladding diameter class Ia graded-index multimode optical fibers.

1.4 Definitions

Delete definition 1.4.250. Renumber the definitions that follow.

1.4.250 side stream scrambling: A data scrambling technique, used by 100BASE-T2 to randomize the sequence of transmitted symbols and avoid the presence of spectral lines in the signal spectrum. Synchronization of the scrambler and descrambler of connected PHYs is required prior to operation.

¹EIA publications are available from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (<http://global.ihs.com/>). JEDEC publications are available from JEDEC, 2001 I Street NW, Washington, DC 20006, USA.

²IEC 61753-022-2 is under development and not available at this time. Its anticipated publication date is April 2003. Until the IEC standard is available, users are encouraged to reference ANSI/TIA/EIA-568-B.3-2000 in its place.

³INCITS publications are available from <http://www.incits.org>.

Change definition 1.4.89 to read as follows:

1.4.89 compatibility interfaces: The Medium Dependent Interface (MDI) cable, the Attachment Unit Interface (AUI) branch cable, and the Media Independent Interface (MII); the three points at which hardware compatibility is defined to allow connection of independently designed and manufactured components to a baseband transmission medium. (See IEEE 802.3 Clause 8.) Several hardware points of attachment have been defined by this standard to allow connection of independently designed and manufactured components to the transmission medium. See 1.1.2.2.

Insert the following definitions at the end of 1.4. These definitions will be renumbered in alphabetic order in a future edition of this standard, placed in the proper sequence with other definitions.

1.4.xxx 10 Gigabit Attachment Unit Interface (XAUI): The interface between two 10 Gigabit Extender Sublayers (XGXS) to extend the reach of the XGMII for 10 Gb/s operation. (See IEEE 802.3 Clause 47.)

1.4.xxx 10 Gigabit Media Independent Interface (XGMII): The interface between the Reconciliation Sublayer (RS) and the Physical Coding Sublayer (PCS) for 10 Gb/s operation. (See IEEE 802.3 Clause 46.)

1.4.xxx 10 Gigabit Sixteen-Bit Interface (XSBI): The interface between the Physical Coding Sublayer (PCS) in 10GBASE-R or the WAN Interface Sublayer (WIS) in 10GBASE-W and the Physical Medium Attachment (PMA) sublayer for 10 Gb/s operation. (See IEEE 802.3 Clause 51.)

1.4.xxx anomaly: A discrepancy between the actual and desired characteristics of an item. This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx bit interleaved parity N (BIP-N): A method of error monitoring using even (or odd) parity, such that an N-bit codeword is generated over a specified portion of an input data stream in such a manner that the i-th bit of the codeword provides even (or odd) parity over the i-th bit of all N-bit sequences in the covered portion of the data stream. This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx container: An SDH term that is equivalent to the payload capacity of a synchronous payload envelope. This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx defect: A limited interruption in the ability of an item to perform a required function. This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx fixed stuff: Null or padding octets inserted to compensate for the bandwidth differences between the byte interleaving and the concatenation rules of SONET/SDH. This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx jitter: The deviation from the ideal timing of an event at the mean amplitude of the signal population. Low-frequency deviations are tracked by the clock recovery circuit, and do not directly affect the timing allocations within a bit cell. Jitter that is not tracked by the clock recovery circuit directly affects the timing allocations in a bit cell. Total Jitter (TJ) is composed of Deterministic Jitter (DJ) and Random Jitter (RJ). Deterministic Jitter is characterized by its bounded peak-to-peak value and a non-Gaussian probability density function. Four kinds of Deterministic Jitter are identified by their specific causes: duty cycle distortion, data dependent, sinusoidal, and uncorrelated (to the data). Random Jitter is characterized by a Gaussian distribution. For example, Random Jitter is the peak-to-peak value at approximately 14 times the standard deviation of the Gaussian distribution for a BER of 10E-12, if the jitter population consists of only Gaussian components.

1.4.xxx lane: A bundle of signals that constitutes a logical subset of a point-to-point interconnect. A lane contains enough signals to communicate a quantum of data and/or control information between the two end-points.

1.4.xxx Optical Modulation Amplitude (OMA): The absolute difference between the optical power of a logic one level and the optical power of a logic zero level.

1.4.xxx pointer: See Payload pointer.

1.4.xxx payload pointer: An indicator of the location of the beginning of the synchronous payload envelope. This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx RIN_xOMA: Relative Intensity Noise. Laser noise in dB/Hz with XdB optical return loss, with respect to the optical modulation amplitude.

1.4.xxx rising edge: A rising edge for a differential signal pair, e.g. signal(P,N) is when, signal(P) transitions from logic low to high and signal(N) transitions from logic high to low.

1.4.xxx scrambler: A randomizing mechanism that is used to eliminate long strings of consecutive identical transmitted symbols and avoid the presence of spectral lines in the signal spectrum without changing the data rate. A self-synchronous scrambler is one in which the current state of the scrambler is the prior n bits of the scrambled output. Therefore, the descrambler can acquire the correct state directly from the received stream. A side-stream scrambler is one in which the current state of the scrambler is dependent only on the prior state of the scrambler and not on the transmitted data. Therefore, the descrambler must acquire state either by searching for a state that decodes a known pattern or by agreement to start at a known state in synchronization with the scrambler. A frame-synchronous scrambler is a side-stream scrambler that begins each frame in a known state. This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx Synchronous Payload Envelope (SPE): A 125-microsecond frame structure composed of STS Path Overhead and bandwidth for payload (payload capacity). The equivalent SDH term is Virtual Container (VC). This definition is derived from ANSI T1.416-1999 and ANSI T1.105-1995, which take precedence.

1.4.xxx WAN Interface Sublayer (WIS): Within 10GBASE-W, a sublayer used to couple the Physical Coding Sublayer (PCS) and the Physical Medium Attachment (PMA) sublayer. The WIS contains functions to perform SONET STS-192c/SDH VC-4-64c framing and scrambling. (See IEEE 802.3 Clause 50.)

1.5 Abbreviations

Insert the following abbreviations in alphabetic order into the abbreviations list in 1.5:

<u>AIS</u>	<u>Alarm Indication Signal</u>
<u>BERT</u>	<u>Bit Error Ratio Tester</u>
<u>BIP</u>	<u>Bit Interleaved Parity</u>
<u>CDR</u>	<u>Clock And Data Recovery Circuit</u>
<u>CID</u>	<u>Consecutive Identical Digit</u>
<u>CJPAT</u>	<u>Continuous Jitter Test Pattern</u>
<u>CRPAT</u>	<u>Continuous Random Test Pattern</u>
<u>DCD</u>	<u>Duty Cycle Distortion</u>
<u>DDJ</u>	<u>Data Dependent Jitter</u>
<u>DJ</u>	<u>Deterministic Jitter</u>
<u>ERDI</u>	<u>Enhanced Remote Defect Indication</u>
<u>LCD</u>	<u>Loss Of Code-Group Delineation</u>

SUPERSEDED

IEEE
Std 802.3ae-2002

AMENDMENT TO IEEE Std 802.3

<u>LOF</u>	<u>Loss Of Framing</u>
<u>LOP</u>	<u>Loss Of Pointer</u>
<u>LOS</u>	<u>Loss Of Signal</u>
<u>LVDS</u>	<u>Low-Voltage Differential Signals</u>
<u>MAN</u>	<u>Metropolitan Area Network</u>
<u>MMD</u>	<u>MDIO Manageable Device</u>
<u>OIF</u>	<u>Optical Internetworking Forum</u>
<u>OMA</u>	<u>Optical Modulation Amplitude</u>
<u>PCB</u>	<u>Printed Circuit Board</u>
<u>PIPO</u>	<u>Parallel In Parallel Out</u>
<u>PISO</u>	<u>Parallel In Serial Out</u>
<u>PLL</u>	<u>Phase Locked Loop</u>
<u>PLM</u>	<u>Path Label Mismatch</u>
<u>PRBS</u>	<u>Pseudo Random Bit Sequence</u>
<u>REI</u>	<u>Remote Error Indication</u>
<u>RJ</u>	<u>Random Jitter</u>
<u>SDH</u>	<u>Synchronous Digital Hierarchy</u>
<u>SEF</u>	<u>Severely Errored Frame</u>
<u>SERDES</u>	<u>Serializer And Deserializer Circuit</u>
<u>SES</u>	<u>Severely Errored Second</u>
<u>SIPO</u>	<u>Serial In Parallel Out</u>
<u>SMSR</u>	<u>Side Mode Suppression Ratio</u>
<u>SNR</u>	<u>Signal To Noise Ratio</u>
<u>SONET</u>	<u>Synchronous Optical Network</u>
<u>SPE</u>	<u>Synchronous Payload Envelope</u>
<u>STS</u>	<u>Synchronous Transport Signal</u>
<u>TSS</u>	<u>Test Signal Structure</u>
<u>UI</u>	<u>Unit Interval</u>
<u>VC</u>	<u>Virtual Container</u>
<u>WAN</u>	<u>Wide Area Network</u>
<u>WIS</u>	<u>WAN Interface Sublayer</u>
<u>WWDM</u>	<u>Wide Wavelength Division Multiplexing</u>
<u>XAUI</u>	<u>10 Gigabit Attachment Unit Interface</u>
<u>XGMII</u>	<u>10 Gigabit Media Independent Interface</u>
<u>SGXS</u>	<u>XGMII Extender Sublayer</u>
<u>XS</u>	<u>Extender Sublayer</u>
<u>XSBI</u>	<u>10 Gigabit Sixteen-Bit Interface</u>

2. Media Access Control (MAC) service specification

2.1 Scope and field of application

Change 2.1 to read as follows:

This clause specifies the services provided by the Media Access Control (MAC) sublayer and the optional MAC Control sublayer to the client of the MAC (~~MAC client~~) (see ~~Figure 2-1~~ see Figure 1-1). MAC clients may include the Logical Link Control (LLC) sublayer, Bridge Relay Entity, or other users of ISO/IEC LAN International Standard MAC services (see ~~Figure 2-22-1~~). The services are described in an abstract way and do not imply any particular implementation, or any exposed interface. There is not necessarily a one-to-one correspondence between the primitives and the formal procedures and interfaces described in Clause 4 and Clause 31.

Delete Figure 2-1.

Renumber the remaining figures (i.e., “Figure 2-2” becomes “Figure 2-1”).

2.3.1.2 Semantics of the service primitive

Change 2.3.1.2 to read as follows:

The semantics of the primitive are as follows:

```
MA_DATA.request    (
                    destination_address,
                    source_address,
m_sdu,
                    mac service data unit,
                    frame check sequence
                    service_class
                    )
```

The destination_address parameter may specify either an individual or a group MAC entity address. It must contain sufficient information to create the DA field that is prepended to the frame by the local MAC sublayer entity and any physical information. The source_address parameter, if present, must specify an individual MAC address. If the source_address parameter is omitted, the local MAC sublayer entity will insert a value associated with that entity. ~~The m_sdu mac service data unit parameter specifies the MAC service data unit to be transmitted by the MAC sublayer entity. There is sufficient information associated with m_sdu the mac service data unit for the MAC sublayer entity to determine the length of the data unit. The frame check sequence parameter, if present, must specify the frame check sequence field for the frame (see 3.2.8). If the frame check sequence parameter is omitted, the local MAC sublayer entity will compute this field and append it to the end of the frame. The service_class parameter indicates a quality of service requested by the MAC client (see 2.3.1.5).~~

2.3.1.5 Additional comments

Change 2.3.1.5 to read as follows:

If this primitive contains the frame check sequence parameter, the MAC client entity must take into account this parameter's special bit-transmission order requirements, as specified in 3.3.

The mapping between the MA_UNITDATA.request primitive specified in ISO/IEC 15802-1 (for end stations) and the MA_DATA.request primitive specified here is as follows:

- a) The user priority parameter specified for MA_UNITDATA.request is not relevant for IEEE 802.3 operation and is ignored by MA_DATA.request.
- b) The access priority parameter specified for MA_UNITDATA.request is not relevant for IEEE 802.3 operation and is ignored by MA_DATA.request.
- c) The frame check sequence parameter is not present for MA_UNITDATA.request.

The mapping between the M_UNITDATA.request primitive specified in ISO/IEC 15802-3 (for MAC Bridges) and the MA_DATA.request primitive specified here is as follows:

- a) The frame type parameter specified for M_UNITDATA.request is not relevant for IEEE 802.3 operation and is ignored by MA_DATA.request.
- b) The mac_action parameter specified for M_UNITDATA.request is not relevant for IEEE 802.3 operation and is ignored by MA_DATA.request.
- c) The user priority parameter specified for M_UNITDATA.request is not relevant for IEEE 802.3 operation and is ignored by MA_DATA.request.
- d) The access priority parameter specified for M_UNITDATA.request is not relevant for IEEE 802.3 operation and is ignored by MA_DATA.request.

The CSMA/CD MAC protocol provides a single quality of service regardless of the service_class requested.

2.3.2.2 Semantics of the service primitive

Change 2.3.2.2 to read as follows:

The semantics of the primitive are as follows:

MA_DATA.indication	(destination_address, source_address, ~~m_sdu~~ , mac_service_data_unit, frame_check_sequence, reception_status)
--------------------	--

The destination_address parameter may be either an individual or a group address as specified by the DA field of the incoming frame. The source_address parameter is an individual address as specified by the SA field of the incoming frame. The ~~m_sdu~~ mac_service_data_unit parameter specifies the MAC service data unit as received by the local MAC entity. The frame_check_sequence parameter is the cyclic redundancy check value (see 3.2.8) as specified by the FCS field of the incoming frame. This parameter may be either omitted or (optionally) passed by the MAC sublayer entity to the MAC client. The reception_status parameter is used to pass status information to the MAC client entity.

2.3.2.5 Additional comments

Change 2.3.2.5 to read as follows:

If the local MAC sublayer entity is designated by the `destination_address` parameter of an `MA_DATA.request`, the indication primitive will also be invoked by the MAC entity to the MAC client entity. This characteristic of the MAC sublayer may be due to unique functionality within the MAC sublayer or characteristics of the lower layers (for example, all frames transmitted to the broadcast address will invoke `MA_DATA.indication` at all stations in the network including the station that generated the request).

If this primitive contains the `frame_check_sequence` parameter, the MAC client entity must take into account this parameter's special bit-transmission order requirements, as specified in 3.3.

The mapping between the `MA_DATA.indication` primitive specified here and the `MA_UNITDATA.indication` primitive specified in ISO/IEC 15802-1 (for end stations) is as follows:

- a) The `user_priority` parameter specified for `MA_UNITDATA.indication` is not relevant for IEEE 802.3 operation.
- b) The `frame_check_sequence` parameter is not present for `MA_UNITDATA.indication`.
- c) The `reception_status` parameter is not mapped to any parameter and is ignored by `MA_UNITDATA.indication`.

The mapping between the `MA_DATA.indication` primitive and the `M_UNITDATA.indication` primitive specified in ISO/IEC 15802-3 (for MAC Bridges) is as follows:

- a) The `frame_type` parameter specified for `M_UNITDATA.indication` is not relevant for IEEE 802.3 operation and is always assigned the value of `user_data_frame`.
- b) The `mac_action` parameter specified for `M_UNITDATA.indication` is not relevant for IEEE 802.3 operation and is always assigned the value of `request_with_no_response`.
- c) The `user_priority` parameter specified for `M_UNITDATA.indication` is not relevant for IEEE 802.3 operation.
- d) The `reception_status` parameter is not mapped to any parameter and is ignored by `M_UNITDATA.indication`.

SUPERSEDED

4. Media Access Control

4.1.2 CSMA/CD operation

Change the first paragraph in 4.1.2 to read as follows:

This subclause provides an overview of frame transmission and reception in terms of the functional model of the architecture. This overview is descriptive; rather than definitional; the formal specifications of the operations described here are given in 4.2 and 4.3. Specific implementations for CSMA/CD mechanisms that meet this standard are given in 4.4. ~~Figure 4-1~~ Figure 1-1 provides the architectural model described functionally in the subclauses that follow.

Delete Figure 4-1.

Renumber the remaining figures (i.e., “Figure 4-2” becomes “Figure 4-1”).

4.1.2.1.1 Transmission without contention

Change 4.1.2.1.1 to read as follows:

When a MAC client requests the transmission of a frame, the Transmit Data Encapsulation component of the CSMA/CD MAC sublayer constructs the frame from the client-supplied data. It prepends a preamble and a Start Frame Delimiter to the beginning of the frame. Using information provided by the client, the CSMA/CD MAC sublayer also appends a PAD at the end of the MAC information field of sufficient length to ensure that the transmitted frame length satisfies a minimum frame_size requirement (see 4.2.3.3). It also prepends destination and source addresses, the length/type field, and appends a frame check sequence to provide for error detection. If the MAC supports the use of client-supplied frame check sequence values, then it shall use the client-supplied value, when present. If the use of client-supplied frame check sequence values is not supported, or if the client-supplied frame check sequence value is not present, then the MAC shall compute this value. The frame is then handed to the Transmit Media Access Management component in the MAC sublayer for transmission.

In half duplex mode, Transmit Media Access Management attempts to avoid contention with other traffic on the medium by monitoring the carrier sense signal provided by the Physical Layer Signaling (PLS) component and deferring to passing traffic. When the medium is clear, frame transmission is initiated (after a brief interframe delay to provide recovery time for other CSMA/CD MAC sublayers and for the physical medium). The MAC sublayer then provides a serial stream of bits to the Physical Layer for transmission.

~~In half duplex mode, at operating speeds above 100 Mb/s~~ at an operating speed of 1000 Mb/s, the minimum frame size is insufficient to ensure the proper operation of the CSMA/CD protocol for the desired network topologies. To circumvent this problem, the MAC sublayer will append a sequence of extension bits to frames which are less than slotTime bits in length so that the duration of the resulting transmission is sufficient to ensure proper operation of the CSMA/CD protocol.

~~In half duplex mode, at operating speeds above 100 Mb/s~~ at an operating speed of 1000 Mb/s, the CSMA/CD MAC may optionally transmit additional frames without relinquishing control of the transmission medium, up to a specified limit.

In full duplex mode, there is no need for Transmit Media Access Management to avoid contention with other traffic on the medium. Frame transmission may be initiated after the interframe delay, regardless of the presence of receive activity. In full duplex mode, the MAC sublayer does not perform either carrier extension or frame bursting.

The Physical Layer performs the task of generating the signals on the medium that represent the bits of the frame. Simultaneously, it monitors the medium and generates the collision detect signal, which in the contention-free case under discussion, remains off for the duration of the frame. A functional description of the Physical Layer is given in Clause 7 and beyond.

When transmission has completed without contention, the CSMA/CD MAC sublayer so informs the MAC client and awaits the next request for frame transmission.

4.1.2.1.2 Reception without contention

Change 4.1.2.1.2 to read as follows:

At each receiving station, the arrival of a frame is first detected by the Physical Layer, which responds by synchronizing with the incoming preamble, and by turning on the receiveDataValid signal. As the encoded bits arrive from the medium, they are decoded and translated back into binary data. The Physical Layer passes subsequent bits up to the MAC sublayer, where the leading bits are discarded, up to and including the end of the preamble and Start Frame Delimiter.

Meanwhile, the Receive Media Access Management component of the MAC sublayer, having observed receiveDataValid, has been waiting for the incoming bits to be delivered. Receive Media Access Management collects bits from the Physical Layer entity as long as the receiveDataValid signal remains on. When the receiveDataValid signal is removed, the frame is truncated to an octet boundary, if necessary, and passed to Receive Data Decapsulation for processing.

Receive Data Decapsulation checks the frame's Destination Address field to decide whether the frame should be received by this station. If so, it passes the Destination Address (DA), the Source Address (SA), the Length/Type-Type/Length field, and the Data and (optionally) the Frame Check Sequence (FCS) fields to the MAC client, along with an appropriate status code, as defined in 4.3.2, indicating reception complete or reception too long. It also checks for invalid MAC frames by inspecting the frame check sequence to detect any damage to the frame enroute, and by checking for proper octet-boundary alignment of the end of the frame. Frames with a valid FCS may also be checked for proper octet-boundary alignment.

In half duplex mode, at operating speeds above 100 Mb/s at an operating speed of 1000 Mb/s, frames may be extended by the transmitting station under the conditions described in 4.2.3.4. The extension is discarded by the MAC sublayer of the receiving station, as defined in the procedural model in 4.2.9.

4.1.4 CSMA/CD access method functional capabilities

Change the first sentence in the first paragraph in 4.1.1 to read as follows:

The following summary of the functional capabilities of the CSMA/CD MAC sublayer is intended as a quick reference guide to the capabilities of the standard, as shown in ~~Figure 4-2~~ Figure 4-1:

Change the following list items at the end of 4.1.4:

- m) Appends extension bits to the first (or only) frame of a burst if it is less than slotTime bits in length when in half duplex mode at speeds above 100 Mb/s at an operating speed of 1000 Mb/s.
- n) Strips extension bits from received frames when in half duplex mode at speeds above 100 Mb/s at an operating speed of 1000 Mb/s.

4.2.2.3 Organization of the procedural model

Change 4.2.2.3 to read as follows:

The procedural model used here is based on seven cooperating concurrent processes. The Frame Transmitter process and the Frame Receiver process are provided by the clients of the MAC sublayer (which may include the LLC sublayer) and make use of the interface operations provided by the MAC sublayer. The other five processes are defined to reside in the MAC sublayer. The seven processes are as follows:

- a) Frame Transmitter process
- b) Frame Receiver process
- c) Bit Transmitter process
- d) Bit Receiver process
- e) Deference process
- f) BurstTime process
- g) SetExtending process

This organization of the model is illustrated in ~~Figure 4-3~~ Figure 4-2 and reflects the fact that the communication of entire frames is initiated by the client of the MAC sublayer, while the timing of collision backoff and of individual bit transfers is based on interactions between the MAC sublayer and the Physical-Layer-dependent bit time.

~~Figure 4-3~~ Figure 4-2 depicts the static structure of the procedural model, showing how the various processes and procedures interact by invoking each other. ~~Figures 4-4a), 4-4b), and 4-5~~ Figures 4-3a, 4-3b, 4-4a and 4-4b summarize the dynamic behavior of the model during transmission and reception, focusing on the steps that shall be performed, rather than the procedural structure that performs them. The usage of the shared_state variables is not depicted in the figures, but is described in the comments and prose in the following subclauses.

Change Figure 4-4a as follows:

- ***Change the figure number from “Figure 4-4” to “Figure 4-3.”***
- ***Change the second note to read as follows:***
 - * Applicable only to half duplex operation at ~~100~~ 1000 Mb/s.

Change Figure 4-4b as follows:

- ***Change the figure number from “Figure 4-4” to “Figure 4-3.”***

Change Figure 4-5a as follows:

- ***Change the figure number from “Figure 4-5” to “Figure 4-4.”***
- ***Change the second note to read as follows:***
 - * Applicable only to half duplex operation at ~~100~~ 1000 Mb/s.

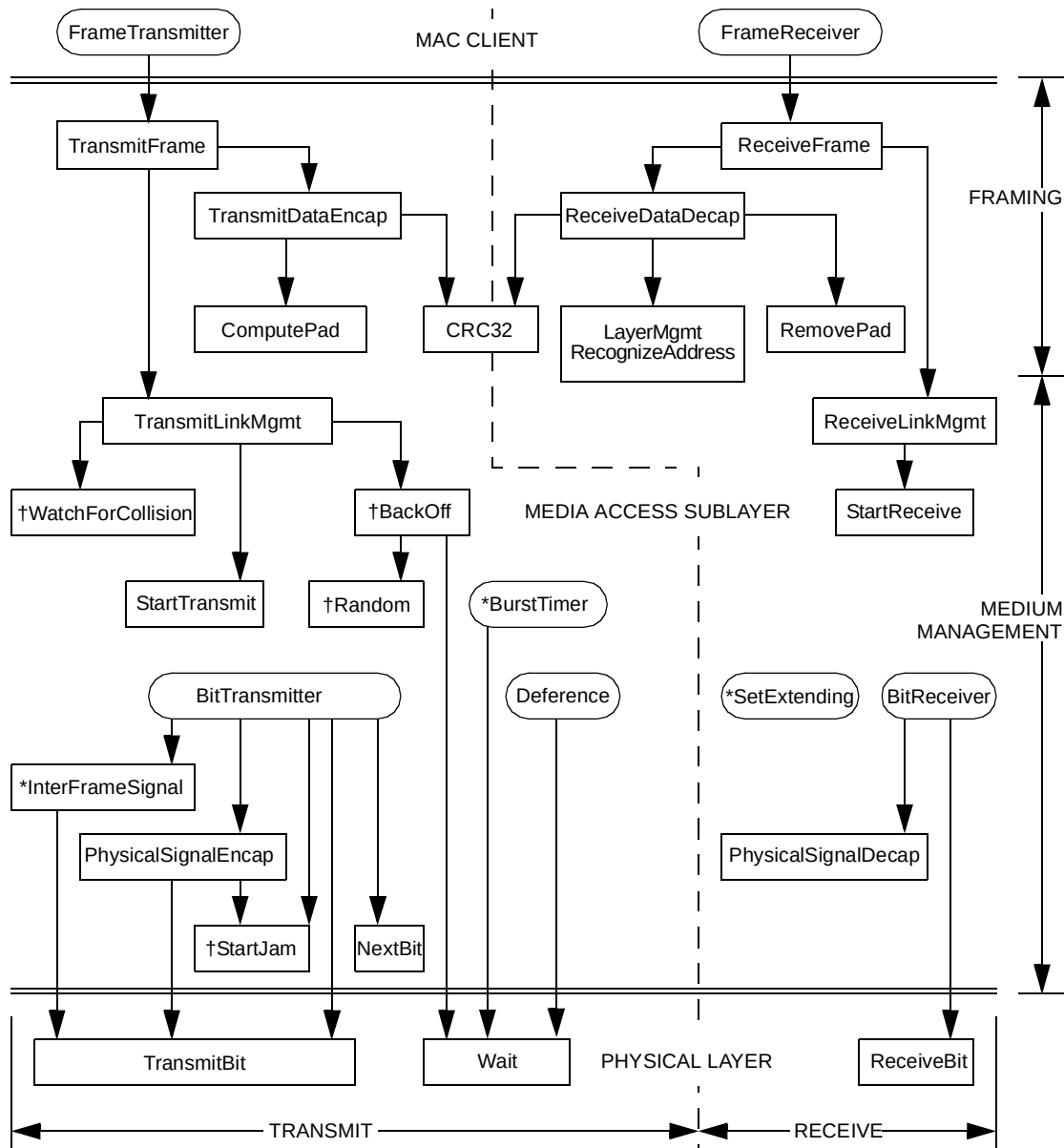
Change Figure 4-5b as follows:

- ***Change the figure number from “Figure 4-5” to “Figure 4-4.”***
- ***Change the note to read as follows:***
 - * Applicable only to half duplex operation at ~~100~~ 1000 Mb/s.

Change Figure 4–3 as follows:

- Change the figure number from “Figure 4–3” to “Figure 4–2.”
- Replace “RecognizeAddress” with “LayerMgmtRecognizeAddress.”
- Add an arrow between Deference and Wait.
- Delete function RealTimeDelay.
- Change the second note to read as follows:

* Applicable only to half duplex operation at $\geq 100,1000$ Mb/s.



† Not applicable to full duplex operation.

* Applicable only to half duplex operation at $\geq 100,1000$ Mb/s.

Figure 4–2—Relationship among CSMA/CD procedures

4.2.2.4 Layer management extensions to procedural model

Change 4.2.2.4 to read as follows:

In order to incorporate network management functions, this Procedural Model has been expanded beyond that provided in ISO/IEC 8802-3: 1990. Network management functions have been incorporated in two ways. First, 4.2.7 through 4.2.10, 4.3.2, and ~~Figure 4-4~~ Figures 4-3a and 4-3b, have been modified and expanded to provide management services. Second, Layer Management procedures have been added as 5.2.4. Note that Pascal variables are shared between Clause 4 and Clause 5. Within the Pascal descriptions provided in Clause 4, a “‡” in the left margin indicates a line that has been added to support management services. These lines are only required if Layer Management is being implemented. These changes do not affect any aspect of the MAC behavior as observed at the LLC-MAC and MAC-PLS interfaces of ISO/IEC 8802-3: 1990.

The Pascal procedural specification shall be consulted for the definitive statement when there is any apparent ambiguity concerning the definition of some aspect of the CSMA/CD MAC access method.

The Layer Management facilities provided by the CSMA/CD MAC and Physical Layer management definitions provide the ability to manipulate management counters and initiate actions within the layers. The managed objects within this standard are defined as sets of attributes, actions, notifications, and behaviors in accordance with IEEE Std 802.1F-1993, and ISO/IEC International Standards for network management.

4.2.3 Frame transmission model

Change 4.2.3 to read as follows:

Frame transmission includes the following data encapsulation and Media Access management aspects:

- a) Transmit Data Encapsulation includes the assembly of the outgoing frame (from the values provided by the MAC client) and frame check sequence generation (if not provided by the MAC client).
- b) Transmit Media Access Management includes carrier deference, interframe spacing, collision detection and enforcement, collision backoff and retransmission, carrier extension and frame bursting.

4.2.3.1 Transmit data encapsulation

Change 4.2.3.1 to read as follows:

4.2.3.1.1 Frame assembly

~~The fields of the CSMA/CD MAC frame are set to the values provided by the MAC client as arguments to the TransmitFrame operation (see 4.3) with the following exceptions: the padding field, the extension field, and the frame check sequence. The padding field is necessary to enforce the minimum frame size. The extension field is necessary to enforce the minimum carrier event duration on the medium in the half duplex mode at operating speeds above 100 Mb/s. The frame check sequence is set to the CRC value generated by the MAC sublayer.~~

4.2.3.1.2 ~~Frame check sequence generation~~

~~The CRC value defined in 3.8 is generated and inserted in the frame check sequence field, following the fields supplied by the MAC client.~~

The fields of the CSMA/CD MAC frame are set to the values provided by the MAC client as arguments to the TransmitFrame operation (see 4.3) with the following possible exceptions: the padding field, the extension field, and the frame check sequence. The padding field is necessary to enforce the minimum frame size. The extension field is necessary to enforce the minimum carrier event duration on the medium in half duplex mode at an operating speed of 1000 Mb/s. The frame check sequence field may be (optionally) provided as an argument to the MAC sublayer. It is optional for a MAC to support the provision of the frame check sequence in such an argument. If this field is provided by the MAC client, the padding field shall also be provided by the MAC client, if necessary. If this field is not provided by the MAC client, or if the MAC does not support the provision of the frame check sequence as an external argument, it is set to the CRC value generated by the MAC sublayer, after appending the padding field, if necessary.

4.2.3.2.2 Interframe spacing

Change 4.2.3.2.2 to read as follows:

As defined in ~~4.2.3.1.1~~ 4.2.3.2.1, the rules for deferring to passing frames ensure a minimum interframe spacing of ~~interFrameSpacing—seconds~~ bit times. This is intended to provide interframe recovery time for other CSMA/CD sublayers and for the physical medium.

Note that interFrameSpacing is the minimum value of the interframe spacing. If necessary for implementation reasons, a transmitting sublayer may use a larger value with a resulting decrease in its throughput. The larger value is determined by the parameters of the implementation, see 4.4.

A larger value for interframe spacing is used for dynamically adapting the nominal data rate of the MAC sublayer to SONET/SDH data rates (with packet granularity) for WAN-compatible applications of this standard. While in this optional mode of operation, the MAC sublayer counts the number of bits sent during a frame's transmission. After the frame's transmission has been completed, the MAC sublayer extends the minimum interframe spacing by a number of bits that is proportional to the length of the previously transmitted frame. For more details, see 4.2.7 and 4.2.8.

4.2.3.2.7 Frame bursting (half duplex mode only)

Change 4.2.3.2.7 to read as follows:

~~At operating speeds above 100 Mb/s~~ an operating speed of 1000 Mb/s, an implementation may optionally transmit a series of frames without relinquishing control of the transmission medium. This mode of operation is referred to as *burst mode*. Once a frame has been successfully transmitted, the transmitting station can begin transmission of another frame without contending for the medium, because all of the other stations on the network will continue to defer to its transmission, provided that it does not allow the medium to assume an idle condition between frames. The transmitting station fills the interframe spacing interval with extension bits, which are readily distinguished from data bits at the receiving stations, and which maintain the detection of carrier in the receiving stations. The transmitting station is allowed to initiate frame transmission until a specified limit, referred to as burstLimit, is reached. The value of burstLimit is specified in ~~4.4.2.4~~ 4.4.2. ~~Figure 4-6~~ Figure 4-5 shows an example of transmission with frame bursting.

The first frame of a burst will be extended, if necessary, as described in 4.2.3.4. Subsequent frames within a burst do not require extension. In a properly configured network, and in the absence of errors, collisions cannot occur during a burst at any time after the first frame of a burst (including any extension) has been transmitted. Therefore, the MAC will treat any collision that occurs after the first frame of a burst, or that occurs after the slotTime has been reached in the first frame of a burst, as a late collision.

Renumber “Figure 4–6” as “Figure 4–5.”

4.2.3.3 Minimum frame size

Change 4.2.3.3 to read as follows:

The CSMA/CD Media Access mechanism requires that a minimum frame length of minFrameSize bits be transmitted. If frameSize is less than minFrameSize, then the CSMA/CD MAC sublayer shall append extra bits in units of octets (pad), after the end of the MAC client data field but prior to calculating, and appending, the FCS (if not provided by the MAC client). The number of extra bits shall be sufficient to ensure that the frame, from the DA field through the FCS field inclusive, is at least minFrameSize bits. If the FCS is (optionally) provided by the MAC client, the pad shall also be provided by the MAC client. The content of the pad is unspecified.

4.2.3.4 Carrier extension (half duplex mode only)

Change 4.2.3.4 to read as follows:

~~At operating speeds above 100 Mb/s~~ an operating speed of 1000 Mb/s, the slotTime employed at slower speeds is inadequate to accommodate network topologies of the desired physical extent. Carrier Extension provides a means by which the slotTime can be increased to a sufficient value for the desired topologies, without increasing the minFrameSize parameter, as this would have deleterious effects. Non-data bits, referred to as extension bits, are appended to frames that are less than slotTime bits in length, so that the resulting transmission is at least one slotTime in duration. Carrier Extension can be performed only if the underlying physical layer is capable of sending and receiving symbols that are readily distinguished from data symbols, as is the case in most physical layers that use a block encoding/decoding scheme. The maximum length of the extension is equal to the quantity (slotTime – minFrameSize). ~~Figure 4–7~~ Figure 4–6 depicts a frame with carrier extension.

The MAC continues to monitor the medium for collisions while it is transmitting extension bits, and it will treat any collision that occurs after the threshold (slotTime) as a late collision.

Renumber “Figure 4–7” as “Figure 4–6.”

4.2.4.1.1 Address recognition

Add the following paragraph to the end of 4.2.4.1.1 to read as follows:

The MAC sublayer may also provide the capability of operating in the promiscuous receive mode. In this mode of operation, the MAC sublayer recognizes and accepts all valid frames, regardless of their Destination Address field values.

4.2.4.2.1 Framing

Change the last paragraph in 4.2.4.2.1 to read as follows:

When a burst of frames is received while operating in half duplex mode at ~~speeds above 100 Mb/s~~ an operating speed of 1000 Mb/s, the individual frames within the burst are delimited by sequences of interframe fill symbols, which are conveyed to the receiving MAC sublayer as extension bits. Once the collision filtering requirements for a given frame, as described in 4.2.4.2.2, have been satisfied, the receipt of an extension bit can be used as an indication that all of the data bits of the frame have been received.

4.2.7 Global declarations

Change 4.2.7 and its related subclauses to read as follows:

This subclause provides detailed formal specifications for the CSMA/CD MAC sublayer. It is a specification of generic features and parameters to be used in systems implementing this media access method. Subclause 4.4 provides values for these sets of parameters for recommended implementations of this media access mechanism.

4.2.7.1 Common constants, types, and variables

The following declarations of constants, types and variables are used by the frame transmission and reception sections of each CSMA/CD sublayer:

```
const
  addressSize = ...; {48 bits in compliance with 3.2.3}
  addressSize = 48; {In bits, in compliance with 3.2.3}
  lengthOrTypeSize = 16; {In bits}
  clientDataSize = ...; {in bits, MAC client Data, see 4.2.2.2, (1)(c)}
  clientDataSize = ...; {In bits, size of MAC client data; see 4.2.2.2, a) 3)}
  padSize = ...; {In bits, = max (0, minFrameSize – (2 x addressSize + lengthOrTypeSize +
                                                                    clientDataSize + crcSize))}

  dataSize = ...; {In bits, = clientDataSize + padSize}
  crcSize = 32; {In bits, 32-bit CRC = 4 octets}
  frameSize = ...; {in bits, = 2 x addressSize + lengthOrTypeSize + dataSize + crcSize, see 4.2.2.2, (1)}
  frameSize = ...; {In bits, = 2 x addressSize + lengthOrTypeSize + dataSize + crcSize; see 4.2.2.2, a)}
  minFrameSize = ...; {In bits, implementation-dependent, see 4.4}
  maxUntaggedFrameSize = ...; {In octets, implementation-dependent, see 4.4}
  qTagPrefixSize = 4; {In octets, length of QTag Prefix, see 3.5}
  extend = ...; {Boolean, true if (slotTime – minFrameSize) > 0, false otherwise}
  extensionBit = ...; {a A nondata value which is used for carrier extension and interframe during bursts}
  extensionErrorBit = ...; {a A nondata value which is used to jam during carrier extension}
  minTypeValue = 1536; {m Minimum value of the Length/Type field for Type interpretation}
  maxValidFrame = maxUntaggedFrameSize – (2 x addressSize + lengthOrTypeSize + crcSize) / 8;
                  {In octets, the maximum length of the MAC client data field. This constant is
                   defined for editorial convenience, as a function of other constants}
  slotTime = ...; {In bit times, unit of time for collision handling, implementation-dependent, see 4.4}
  preambleSize = ...; {56 bits, see 4.2.5}
  preambleSize = 56; {In bits, see 4.2.5}
  sfdSize = 8; {8-bit In bits, start frame delimiter}
  headerSize = ...; {64 bits, sum of preambleSize and sfdSize}
  headerSize = 64; {In bits, sum of preambleSize and sfdSize}

type
  Bit = 0..1(0, 1);
  PhysicalBit = 0, 1, extensionBit, extensionErrorBit(0, 1, extensionBit, extensionErrorBit);
```

```

        {bBits transmitted to the Physical Layer can be either 0, 1, extensionBit or
        extensionErrorBit. Bits received from the Physical Layer can be either 0, 1
        or extensionBit-}
AddressValue = array [1..addressSize] of Bit;
LengthOrTypeValue = array [1..lengthOrTypeSize] of Bit;
DataValue = array [1..dataSize] of Bit; {Contains the portion of the frame that starts with the first bit
        following the Length/Type field and ends with the last bit
        prior to the FCS field. For VLAN Tagged frames, this value
        includes the Tag Control Information field and the original
        MAC client Length/Type field. See 3.5}
CRCValue = array [1..crcSize] of Bit;
PreambleValue = array [1..preambleSize] of Bit;
SfdValue = array [1..sfdSize] of Bit;
ViewPoint = (fields, bits); {Two ways to view the contents of a frame}
HeaderViewPoint = (headerFields, headerBits);
Frame = record {Format of Media Access frame}
    case view: ViewPoint of
        fields: (
            destinationField: AddressValue;
            sourceField: AddressValue;
            lengthOrTypeField: LengthOrTypeValue;
            dataField: DataValue;
            fcsField: CRCValue);
        bits: (contents: array [1..frameSize] of Bit)
    end; {Frame}

Header = record {Format of preamble and start frame delimiter}
    case headerView-: HeaderViewPoint of
        headerFields-: (
            preamble-: PreambleValue;
            sfd-: SfdValue);
        headerBits-:-(
            headerContents: array [1..headerSize] of Bit)
        headerBits: (headerContents: array [1..headerSize] of Bit)
    end; {dDefines header for MAC frame}

var
    halfDuplex: Boolean; {Indicates the desired mode of operation. halfDuplex is a static variable; its value
        shall only be changed by the invocation of the Initialize procedure}
        its value does not change between invocations of the Initialize procedure}
    
```

4.2.7.2 Transmit state variables

The following items are specific to frame transmission. (See also 4.4.)

```

const
    interFrameSpacing = ...-; {In bit times, minimum time gap between frames. Equal to interFrameGap,
        see 4.4}
    interFrameSpacingPart1 = ...-; {In bit times, duration of the first portion of interFrameSpacing. In the
        range of 0 up to 2/3 of interFrameSpacing}
    interFrameSpacingPart2 = ...-; {In bit times, duration of the remainder of interFrameSpacing. Equal to
        interFrameSpacing – interFrameSpacingPart1}
    interFrameSize = ...-; {in bits, length of interframe fill during a burst. Equal to interFrameGap
        divided by the bit period}
    
```

ifsStretchRatio = ...; {In bits, determines the number of bits in a frame that require one octet of interFrameSpacing extension, when ifsStretchMode is enabled; implementation dependent, see 4.4}

attemptLimit = ...; {Max number of times to attempt transmission}

backOffLimit = ...; {Limit on number of times to back off}

burstLimit = ...; ~~{In bits, limit for initiation of frame transmission in Burst Mode, implementation dependent, see 4.4}~~

jamSize = ...; ~~{In bits, the value depends upon medium and collision detect implementation}~~

var

outgoingFrame: Frame; {The frame to be transmitted}

outgoingHeader: Header;

currentTransmitBit, lastTransmitBit: 1..frameSize; {Positions of current and last outgoing bits in outgoingFrame}

lastHeaderBit: 1..headerSize;

deferring: Boolean; {Implies any pending transmission must wait for the medium to clear}

frameWaiting: Boolean; {Indicates that outgoingFrame is deferring}

attempts: 0..attemptLimit; {Number of transmission attempts on outgoingFrame}

newCollision: Boolean; {Indicates that a collision has occurred but has not yet been jammed}

transmitSucceeding: Boolean; {Running indicator of whether transmission is succeeding}

burstMode: Boolean; {Indicates the desired mode of operation, and enables the transmission of multiple frames in a single carrier event. burstMode is a static variable; its ~~value does not change between invocations of the Initialize procedure.~~ value shall only be changed by the invocation of the Initialize procedure}

bursting: Boolean; {In burstMode, the given station has acquired the medium and the burst timer has not yet expired}

burstStart: Boolean; {In burstMode, indicates that the first frame transmission is in progress}

extendError: Boolean; {Indicates a collision occurred while sending extension bits}

ifsStretchMode: Boolean; {Indicates the desired mode of operation, and enables the lowering of the average data rate of the MAC sublayer (with frame granularity), using extension of the minimum interFrameSpacing. ifsStretchMode is a static variable; its value shall only be changed by the invocation of the Initialize procedure}

ifsStretchCount: 0..ifsStretchRatio; {In bits, a running counter that counts the number of bits during a frame's transmission that are to be considered for the minimum interFrameSpacing extension, while operating in ifsStretchMode}

ifsStretchSize: 0..(((maxUntaggedFrameSize + qTagPrefixSize) x 8 + headerSize + interFrameSpacing + ifsStretchRatio - 1) div ifsStretchRatio);
{In octets, a running counter that counts the integer number of octets that are to be added to the minimum interFrameSpacing, while operating in ifsStretchMode}

4.2.7.3 Receive state variables

The following items are specific to frame reception. (See also 4.4.)

var

incomingFrame: Frame; {The frame being received}

receiving: Boolean; {Indicates that a frame reception is in progress}

excessBits: 0..7; {Count of excess trailing bits beyond octet boundary}

receiveSucceeding: Boolean; {Running indicator of whether reception is succeeding}

validLength: Boolean; {Indicator of whether received frame has a length error}

exceedsMaxLength: Boolean; {Indicator of whether received frame has a length longer than the maximum permitted length}

extending: Boolean; {Indicates whether the current frame is subject to carrier extension}

extensionOK: Boolean; {Indicates whether any bit errors were found in the extension part of a frame,

which is not checked by the CRC}

passReceiveFCSType: Boolean; {Indicates the desired mode of operation, and enables passing of the frame check sequence field of all received frames from the MAC sublayer to the MAC client. passReceiveFCSType is a static variable}

4.2.7.4 Summary of interlayer interfaces

- a) The interface to the MAC client, defined in 4.3.2, is summarized below:

type

TransmitStatus = (transmitDisabled, transmitOK, excessiveCollisionError, lateCollisionErrorStatus);
 {Result of TransmitFrame operation, reporting of lateCollisionErrorStatus is optional for MACs operating at speeds at or below 100Mb/s}
 ReceiveStatus = (receiveDisabled, receiveOK, frameTooLong, frameCheckError, lengthError, alignmentError); {Result of ReceiveFrame operation}

function TransmitFrame (

destinationParam: AddressValue;
 sourceParam: AddressValue;
 lengthOrTypeParam: LengthOrTypeValue;
~~dataParam: DataValue): TransmitStatus; {Transmits one frame}~~
dataParam: DataValue;
fcsParamValue: CRCValue;
fcsParamPresent: Bit): TransmitStatus; {Transmits one frame}

function ReceiveFrame (

var destinationParam: AddressValue;
 var sourceParam: AddressValue;
 var lengthOrTypeParam: LengthOrTypeValue;
~~var dataParam: DataValue): ReceiveStatus; {Receives one frame}~~
var dataParam: DataValue;
var fcsParamValue: CRCValue;
var fcsParamPresent: Bit): ReceiveStatus; {Receives one frame}

- b) The interface to the Physical Layer, defined in 4.3.3, is summarized in the following:

var

receiveDataValid: Boolean; {Indicates incoming bits}
 carrierSense: Boolean; {In half duplex mode, indicates that transmission should defer}
 transmitting: Boolean; {Indicates outgoing bits}
 collisionDetect: Boolean; {Indicates medium contention}

procedure TransmitBit (bitParam: PhysicalBit); {Transmits one bit}

function ReceiveBit: PhysicalBit; {Receives one bit}

procedure Wait (bitTimes: integer); {Waits for indicated number of bit times}

4.2.7.5 State variable initialization

The procedure Initialize must be run when the MAC sublayer begins operation, before any of the processes begin execution. Initialize sets certain crucial shared state variables to their initial values. (All other global variables are appropriately reinitialized before each use.) Initialize then waits for the medium to be idle, and starts operation of the various processes.

NOTE—Care should be taken to ensure that the time from the completion of the Initialize process to when the first packet transmission begins is at least an interFrameGap.

If Layer Management is implemented, the Initialize procedure shall only be called as the result of the initializeMAC action (30.3.1.2.1).

```

procedure Initialize;
beginbegin
    frameWaiting := false;
    deferring := false;
    newCollision := false;
    transmitting := false; {An interface to Physical Layer; see below}
    receiving := false;
    halfDuplex := ...; {True for half duplex operation, false for full duplex operation. For operation at
speeds above 1000 Mb/s, halfDuplex shall always be false} halfDuplex is a static
variable; its value does not change between invocations of the Initialize
    bursting := false;
    burstMode := ...; {True for half duplex operation at speeds above 100 Mb/s an operating speed of 1000
Mb/s, when multiple frames' transmission in a single carrier event is desired and
supported, false otherwise} burstMode is a static variable; its value does not change
between invocations of the Initialize procedure
    extending := extend andand halfDuplex;
    ifsStretchMode := ...; {True for operating speeds above 1000 Mb/s when lowering the average data rate
of the MAC sublayer (with frame granularity) is desired and supported, false
otherwise}
    ifsStretchCount := 0;
    ifsStretchSize := 0;
    passReceiveFCSMode := ...; {True when enabling the passing of the frame check sequence of all
received frames from the MAC sublayer to the MAC client is desired and
supported, false otherwise}
while carrierSense or receiveDataValid do nothing
if halfDuplex then while carrierSense or receiveDataValid do nothing
else while receiveDataValid do nothing
    {Start execution of all processes}
end; {Initialize}

```

4.2.8 Frame transmission

Change subclause 4.2.8 to read as follows:

The algorithms in this subclause define MAC sublayer frame transmission. The function TransmitFrame implements the frame transmission operation provided to the MAC client:

```

function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    lengthOrTypeParam: LengthOrTypeValue;
dataParam: DataValue): TransmitStatus;
dataParam: DataValue;
fcsParamValue: CRCValue;
fcsParamPresent: Bit): TransmitStatus;
procedure TransmitDataEncap; ... {nNested procedure; see body below}
begin
    if transmitEnabled then
        begin
            TransmitDataEncap;

```

```

        TransmitFrame := TransmitLinkMgmt
    end
else TransmitFrame := transmitDisabled
end; {TransmitFrame}

```

If transmission is enabled, TransmitFrame calls the internal procedure TransmitDataEncap to construct the frame. Next, TransmitLinkMgmt is called to perform the actual transmission. The TransmitStatus returned indicates the success or failure of the transmission attempt.

TransmitDataEncap builds the frame and places the 32-bit CRC in the frame check sequence field:

```

procedure TransmitDataEncap;
begin
    with outgoingFrame do
        begin {Assemble frame}
            view := fields;
            destinationField := destinationParam;
            sourceField := sourceParam;
            lengthOrTypeField := lengthOrTypeParam;
            dataField := ComputePad(dataParam);
            fcsField := CRC32(outgoingFrame);
            if fcsParamPresent then
                begin
                    dataField := dataParam; {No need to generate pad if the FCS is passed from MAC client}
                    fcsField := fcsParamValue {Use the FCS passed from MAC client}
                end
            else
                begin
                    dataField := ComputePad(dataParam);
                    fcsField := CRC32(outgoingFrame)
                end;
            view := bits
        end {Assemble frame}
    with outgoingHeader do
        begin
            headerView := headerFields;
            preamble := ...; { * '1010...10,' LSB to MSB* }
            sfd := ...; { * '10101011,' LSB to MSB* }
            headerView := headerBits
        end
    end; {TransmitDataEncap}

```

If the MAC client chooses to generate the frame check sequence field for the frame, it passes this field to the MAC sublayer via the fcsParamValue parameter. If the fcsParamPresent parameter is true, TransmitDataEncap uses the fcsParamValue parameter as the frame check sequence field for the frame. Such a frame shall not require any padding, since it is the responsibility of the MAC client to ensure that the frame meets the minFrameSize constraint. If the fcsParamPresent parameter is false, the fcsParamValue parameter is unspecified. TransmitDataEncap first calls the ComputePad function, followed by a call to the CRC32 function to generate the padding (if necessary) and the frame check sequence field for the frame internally to the MAC sublayer.

ComputePad appends an array of arbitrary bits to the MAC client data to pad the frame to the minimum frame size;

```

function ComputePad(
    var dataParam: DataValue): DataValue;
begin
    ComputePad := {Append an array of size padSize of arbitrary bits to the MAC client dataField}
end; {ComputePadParam}

function ComputePad(var dataParam: DataValue): DataValue;
begin
    ComputePad := {Append an array of size padSize of arbitrary bits to the MAC client dataField}
end; {ComputePad}

```

TransmitLinkMgmt attempts to transmit the frame. In half duplex mode, it first defers to any passing traffic. In half duplex mode, if a collision occurs, transmission is terminated properly and retransmission is scheduled following a suitable backoff interval:

```

function TransmitLinkMgmt: TransmitStatus;
begin
    attempts := 0;
    transmitSucceeding := false;
    lateCollisionCount := 0;
    deferred := false; {Initialize}
    excessDefer := false;
    while (attempts < attemptLimit) and (not transmitSucceeding)
        and (not extend or lateCollisionCount = 0) do
            {No retransmission after late collision if operating at > 100 1000 Mb/s}
    begin {Loop}
        if bursting then {This is a burst continuation}
            frameWaiting := true {Start transmission without checking deference}
        else {Non bursting case, or first frame of a burst}
            begin
                if attempts > 0 then BackOff;
                if halfDuplex then frameWaiting := true;
                frameWaiting := true;
                while deferring do {Defer to passing frame, if any3}
                    if halfDuplex then deferred := true;
                    begin
                    nothing;
                    if halfDuplex then deferred := true
                    end;
                burstStart := true;
                if burstMode then bursting := true
            end;
            lateCollisionError := false;
            StartTransmit;
            frameWaiting := false;
            if halfDuplex then
                begin
                    frameWaiting := false;
                    while transmitting do WatchForCollision;
                    if lateCollisionError then lateCollisionCount := lateCollisionCount + 1;
                    lateCollisionCount := lateCollisionCount + 1;

```

³ The Deference process ensures that the reception of traffic does not cause deferring to be true when in full duplex mode. Deferring is used in full duplex mode to enforce the minimum interpacket gap spacing.

```

        attempts := attempts + 1
    end {Half duplex mode}
    else while transmitting do nothing {Full duplex mode}
end; {Loop}
LayerMgmtTransmitCounters; {Update transmit and transmit error counters in 5.2.4.2}
if transmitSucceeding then
    begin
        if burstMode then burstStart := false; {Can't be the first frame anymore}
        TransmitLinkMgmt := transmitOK
    end
else if (extend and lateCollisionCount > 0) then TransmitLinkMgmt := lateCollisionErrorStatus;
    TransmitLinkMgmt := lateCollisionErrorStatus;
else TransmitLinkMgmt := excessiveCollisionError
end; {TransmitLinkMgmt}

```

Each time a frame transmission attempt is initiated, StartTransmit is called to alert the BitTransmitter process that bit transmission should begin:

```

procedure StartTransmit;
begin
    currentTransmitBit := 1;
    lastTransmitBit := frameSize;
    transmitSucceeding := true;
    transmitting := true;
    lastHeaderBit := headerSize
end; {StartTransmit}

```

In half duplex mode, TransmitLinkMgmt monitors the medium for contention by repeatedly calling WatchForCollision, once frame transmission has been initiated:

```

procedure WatchForCollision;
begin
    if transmitSucceeding and collisionDetect then
        begin
            if currentTransmitBit > (slotTime – headerSize) then lateCollisionError := true;
                lateCollisionError := true;
            newCollision := true;
            transmitSucceeding := false;
            if burstMode then
                begin
                    bursting := false;
                    if not burstStart then
                        lateCollisionError := true {Every collision is late, unless it hits the first frame in a burst}
                    end
                end
            end
        end
end; {WatchForCollision}

```

WatchForCollision, upon detecting a collision, updates newCollision to ensure proper jamming by the BitTransmitter process. The current transmit bit number is checked to see if this is a late collision. If the collision occurs later than a collision window of slotTime bits into the packet, it is considered as evidence of a late collision. The point at which the collision is received is determined by the network media propagation

time and the delay time through a station and, as such, is implementation-dependent (see 4.1.2.2). While operating at speeds of 100 Mb/s or lower, an implementation may optionally elect to end retransmission attempts after a late collision is detected. While operating at ~~speeds above 100 Mb/s~~ the speed of 1000 Mb/s, an implementation shall end retransmission attempts after a late collision is detected.

After transmission of the jam has been completed, if TransmitLinkMgmt determines that another attempt should be made, BackOff is called to schedule the next attempt to retransmit the frame.

```
function Random (low, high: integer): integer;
begin
    Random := ...{uUniformly distributed random integer  $r_1$  such that  $low \leq r < high$ }
end; {Random}
```

BackOff performs the truncated binary exponential backoff computation and then waits for the selected multiple of the slot time;

```
var maxBackOff: 2..1024; {Working variable of BackOff}
procedure BackOff;
begin
    if attempts = 1 then maxBackOff := 2
    else if attempts ≤ backOffLimit then maxBackOff := maxBackOff x 2;
    Wait(slotTime x Random(0, maxBackOff))
end; {BackOff}
```

BurstTimer is a process that does nothing unless the bursting variable is true. When bursting is true, BurstTimer increments burstCounter until the burstLimit limit is reached, whereupon BurstTimer assigns the value false to bursting;

```
process BurstTimer;
var burstCounter: integer;
begin
    cycle
    while not bursting do nothing; {wait for a burst}
    burstCounter := 0;
    while bursting and (burstCounter < burstLimit) do
        begin
            Wait(1);
            burstCounter := burstCounter + 1
        end;
    bursting := false
end {burstMode cycle}
end; {BurstTimer}
```

```
process BurstTimer;
begin
    cycle
    while not bursting do nothing; {Wait for a burst}
    Wait(burstLimit);
    bursting := false
end {burstMode cycle}
end; {BurstTimer}
```

The Deference process runs asynchronously to continuously compute the proper value for the variable deferring. In the case of half duplex burst mode, deferring remains true throughout the entire burst. Interframe

spacing may be used to lower the average data rate of a MAC at operating speeds above 1000 Mb/s in the full duplex mode, when it is necessary to adapt it to the data rate of a WAN-based physical layer. When interframe stretching is enabled, deferring remains true throughout the entire extended interframe gap, which includes the sum of interFrameSpacing and the interframe extension as determined by the BitTransmitter:

```

process Deference;
  var realTimeCounter: integer; wasTransmitting: Boolean;
begin
  if halfDuplex then cycle {Half duplex loop}
    while not carrierSense do nothing; {Watch for carrier to appear}
    deferring := true; {Delay start of new transmissions}
    wasTransmitting := transmitting;
    while carrierSense or transmitting do wasTransmitting := wasTransmitting or transmitting;
      wasTransmitting := wasTransmitting or transmitting;
    if wasTransmitting then Wait(interFrameSpacingPart1) {Time out first part of interframe gap}
      begin
        StartRealTimeDelay; {time out first part interframe gap}
        while RealTimeDelay(interFrameSpacingPart1) do nothing
        end
      end
    else
      begin
        StartRealTimeDelay;
        repeat
          while carrierSense do StartRealTimeDelay
          until not RealTimeDelay(interFrameSpacingPart1)
          realTimeCounter := interFrameSpacingPart1;
        repeat
          while carrierSense do realTimeCounter := interFrameSpacingPart1;
          Wait(1);
          realTimeCounter := realTimeCounter - 1
        until (realTimeCounter = 0)
        end;
        StartRealTimeDelay; {time out second part interframe gap}
        while RealTimeDelay(interFrameSpacingPart2) do nothing;
        Wait(interFrameSpacingPart2); {Time out second part of interframe gap}
        deferring := false; {Allow new transmissions to proceed}
        while frameWaiting do nothing {Allow waiting transmission, if any}
      end {Half duplex loop}
    else cycle {Full duplex loop}
      while not transmitting do nothing; {Wait for the start of a transmission}
      deferring := true; {Inhibit future transmissions}
      while transmitting do nothing; {Wait for the end of the current transmission}
      StartRealTimeDelay; {time out an interframe gap}
      while RealTimeDelay(interFrameSpacing) do nothing;
      Wait(interFrameSpacing + ifsStretchSize x 8); {Time out entire interframe gap and IFS extension}
      if not frameWaiting then {Don't roll over the remainder into the next frame}
        begin
          Wait(8);
          ifsStretchCount := 0
        end
      end
      deferring := false {Don't inhibit transmission}
    end {Full duplex loop}
  end; {Deference}

```

If the ifsStretchMode is enabled, the Deference process continues to enforce interframe spacing for an additional number of bit times, after the completion of timing the interFrameSpacing. The additional number of bit times is reflected by the variable ifsStretchSize. If the variable ifsStretchCount is less than ifsStretchRatio and the next frame is ready for transmission (variable frameWaiting is true), the Deference process enforces interframe spacing only for the integer number of octets, as indicated by ifsStretchSize, and saves ifsStretchCount for the next frame's transmission. If the next frame is not ready for transmission (variable frameWaiting is false), then the Deference process initializes the ifsStretchCount variable to zero.

```

procedure StartRealTimeDelay
  begin
    {reset the realtime timer and start it timing}
  end; {StartRealTimeDelay}

function RealTimeDelay ( $\mu$ sec:real): Boolean;
  begin
    {return the value true if the specified number of microseconds have
    not elapsed since the most recent invocation of StartRealTimeDelay,
    otherwise return the value false}
  end; {RealTimeDelay}

```

The BitTransmitter process runs asynchronously, transmitting bits at a rate determined by the Physical Layer's TransmitBit operation:

```

process BitTransmitter;
  begin
    cycle {Outer loop}
    if transmitting then
      begin {Inner loop}
        extendError := false;
        if ifsStretchMode then {Calculate the counter values}
        begin
          ifsStretchSize := (ifsStretchCount + headerSize + frameSize + interFrameSpacing) div
            ifsStretchRatio; {Extension of the interframe spacing}
          ifsStretchCount := (ifsStretchCount + headerSize + frameSize + interFrameSpacing)
            mod ifsStretchRatio {Remainder to carry over into the next frame's transmission}
        end;
        PhysicalSignalEncap; {Send preamble and start of frame delimiter}
        while transmitting do
          begin
            if (currentTransmitBit > lastTransmitBit) then TransmitBit(extensionBit)
            else if extendError then TransmitBit(extensionErrorBit) {Jam in extension}
            if extendError then
              TransmitBit(extensionErrorBit) {jam in extension}
            else TransmitBit(outgoingFrame[currentTransmitBit]);
              TransmitBit(outgoingFrame[currentTransmitBit]);
            if newCollision then StartJam else NextBit
          end;
        if bursting then
          begin
            InterFrameSignal;
            if extendError then
              if transmitting then transmitting := false {TransmitFrame may have been
              called during InterFrameSignal}
              {TransmitFrame may have been called during InterFrameSignal}
          end;
        end;
      end;
    end;
  end;

```

```

        else IncLargeCounter(lateCollision);
            {Count late collisions which were missed by TransmitLinkMgmt}
            IncLargeCounter(lateCollision); {count late collisions which
                were missed by TransmitLinkMgmt}
            bursting := bursting and (frameWaiting or transmitting)
        end
    end {iInner loop}
end {oOuter loop}
end; {BitTransmitter}

```

The bits transmitted to the physical layer can take one of four values: data zero (0), data one (1), extension-Bit (EXTEND), or extensionErrorBit (EXTEND_ERROR). The values extensionBit and extensionErrorBit are not transmitted between the first preamble bit of a frame and the last data bit of a frame under any circumstances. The BitTransmitter calls the procedure TransmitBit with bitParam = extensionBit only when it is necessary to perform carrier extension on a frame after all of the data bits of a frame have been transmitted. The BitTransmitter calls the procedure TransmitBit with bitParam = extensionErrorBit only when it is necessary to jam during carrier extension.

```

procedure PhysicalSignalEncap;
begin
    while currentTransmitBit ≤ lastHeaderBit do
        begin
            TransmitBit(outgoingHeader[currentTransmitBit]); {tTransmit header one bit at a time}
            currentTransmitBit := currentTransmitBit + 1
        end;
        if newCollision then StartJam else currentTransmitBit := 1
            currentTransmitBit := 1
        end; {PhysicalSignalEncap}

```

The procedure InterFrameSignal fills the interframe interval between the frames of a burst with extension-Bits. InterFrameSignal also monitors the variable collisionDetect during the interframe interval between the frames of a burst, and will end a burst if a collision occurs during the interframe interval. The procedural model is defined such that a MAC operating in the burstMode will emit an extraneous sequence of interFrameSize extensionBits in the event that there are no additional frames ready for transmission after InterFrameSignal returns. Implementations may be able to avoid sending this extraneous sequence of extensionBits if they have access to information (such as the occupancy of a transmit queue) that is not assumed to be available to the procedural model.

```

procedure InterFrameSignal;
    var interFrameCount, interFrameTotal: integer;
begin
    interFrameCount := 0;
    interFrameTotal := interFrameSize;
    interFrameTotal := interFrameSpacing;
    while interFrameCount < interFrameTotal do
        begin
            if not extendError then TransmitBit(extensionBit)
                TransmitBit(extensionBit)
            else TransmitBit(extensionErrorBit);
                TransmitBit(extensionErrorBit);
            interFrameCount := interFrameCount + 1;
            if collisionDetect and not extendError then
                begin
                    bursting := false;

```

```

        extendError := true;
        interFrameCount := 0;
        interFrameTotal := jamSize
    end
end
end; {InterFrameSignal}

procedure NextBit;
begin
    currentTransmitBit := currentTransmitBit + 1;
    if halfDuplex and burstStart and transmitSucceeding then {eCarrier extension may be required}
        transmitting := (currentTransmitBit ≤ max(lastTransmitBit, slotTime))
    else transmitting := (currentTransmitBit ≤ lastTransmitBit)
        transmitting := (currentTransmitBit ≤ lastTransmitBit)
    end; {NextBit}

procedure StartJam;
begin
    extendError := currentTransmitBit > lastTransmitBit;
    currentTransmitBit := 1;
    lastTransmitBit := jamSize;
    newCollision := false
end; {StartJam}

```

BitTransmitter, upon detecting a new collision, immediately enforces it by calling StartJam to initiate the transmission of the jam. The jam should contain a sufficient number of bits of arbitrary data so that it is assured that both communicating stations detect the collision. (StartJam uses the first set of bits of the frame up to jamSize, merely to simplify this program.)

4.2.9 Frame reception

Change subclause 4.2.9 to read as follows:

The algorithms in this subclause define CSMA/CD Media Access sublayer frame reception.

The function ReceiveFrame implements the frame reception operation provided to the MAC client:

```

function ReceiveFrame (
    var destinationParam: AddressValue;
    var sourceParam: AddressValue;
    var lengthOrTypeParam: LengthOrTypeValue;
    var dataParam: DataValue): ReceiveStatus;
    var dataParam: DataValue;
    var fcsParamValue: CRCValue;
    var fcsParamPresent: Bit): ReceiveStatus;
function ReceiveDataDecap: ReceiveStatus; --- {n} {Nested function; see body below}
begin
    if receiveEnabled then
        repeat
            ReceiveLinkMgmt;
            ReceiveFrame := ReceiveDataDecap;
        until receiveSucceeding
    else ReceiveFrame := receiveDisabled
        ReceiveFrame := receiveDisabled
    end; {ReceiveFrame}

```

If enabled, ReceiveFrame calls ReceiveLinkMgmt to receive the next valid frame, and then calls the internal function ReceiveDataDecap to return the frame's fields to the MAC client if the frame's address indicates that it should do so. The returned ReceiveStatus indicates the presence or absence of detected transmission errors in the frame.

```

function ReceiveDataDecap: ReceiveStatus;
±   var status: ReceiveStatus; {Holds receive status information}
begin
±   with incomingFrame do
±       begin
±           view := fields;
±           receiveSucceeding := RecognizeAddress(incomingFrame, destinationField);
±           receiveSucceeding := LayerMgmtRecognizeAddress(destinationField);
±           if receiveSucceeding then
±               begin {Disassemble frame}
±                   destinationParam := destinationField;
±                   sourceParam := sourceField;
±                   lengthOrTypeParam := lengthOrTypeField;
±                   dataParam := RemovePad(lengthOrTypeField, dataField);
±                   fcsParamValue := fcsField;
±                   fcsParamPresent := passReceiveFCSMode;
±                   exceedsMaxLength := ...; {Check to determine if receive frame size exceeds the maximum
±                                           permitted frame size. MAC implementations may use either
±                                           maxUntaggedFrameSize or (maxUntaggedFrameSize +
±                                           qTagPrefixSize) for the maximum permitted frame size,
±                                           either as a constant or as a function of whether the frame being
±                                           received is a basic or tagged frame (see 3.2, 3.5). In
±                                           implementations that treat this as a constant, it is recommended
±                                           that the larger value be used. The use of the smaller value
±                                           in this case may result in valid tagged frames exceeding the
±                                           maximum permitted frame size;}
±                   if exceedsMaxLength then status := frameTooLong
±                   else if fcsField = CRC32(incomingFrame) and extensionOK then
±                       if validLength then status := receiveOK else status := lengthError
±                       else if excessBits = 0 or not extensionOK then status := frameCheckError
±                       else status := alignmentError;
±                       if fcsField = CRC32(incomingFrame) and extensionOK then
±                           begin
±                               if validLength then status := receiveOK
±                               else status := lengthError
±                           end
±                       else
±                           begin
±                               if excessBits = 0 or not extensionOK then status := frameCheckError
±                               else status := alignmentError
±                           end;
±                       LayerMgmtReceiveCounters(status); {Update receive counters in 5.2.4.3}
±                       {update receive and receive error counters in 5.2.4.3}
±                       view := bits
±                   end {Disassemble frame}
±               end; {With incomingFrame}
±           ReceiveDataDecap := status
±       end; {ReceiveDataDecap}

```

```

function RecognizeAddress (address: AddressValue): Boolean;
begin
    RecognizeAddress := ...; {Returns true for the set of physical, broadcast,
                                and multicast group addresses corresponding
                                to this station}
end; {RecognizeAddress}

function LayerMgmtRecognizeAddress(address: AddressValue): Boolean;
begin
    if {promiscuous receive enabled} then LayerMgmtRecognizeAddress := true;
    if address = ... {MAC station address} then LayerMgmtRecognizeAddress := true;
    if address = ... {Broadcast address} then LayerMgmtRecognizeAddress := true;
    if address = ... {One of the addresses on the multicast list and multicast reception is enabled} then
        LayerMgmtRecognizeAddress := true;
    LayerMgmtRecognizeAddress := false
end; {LayerMgmtRecognizeAddress}

```

The function RemovePad strips any padding that was generated to meet the minFrameSize constraint, if possible. When the MAC sublayer operates in the mode that enables passing of the frame check sequence field of all received frames to the MAC client (passReceiveFCSType variable is true), it shall not strip the padding and it shall leave the data field of the frame intact. Length checking is provided for Length interpretations of the Length/Type field. For Length/Type field values in the range between maxValidFrame and minTypeValue, the behavior of the RemovePad function is unspecified;

```

function RemovePad(var lengthOrTypeParam: LengthOrTypeValue; dataParam: DataValue): DataValue;
var lengthOrTypeParam: LengthOrTypeValue; dataParam: DataValue; DataValue;
begin
    if lengthOrTypeParam ≥ minTypeValue then
        begin
            validLength := true; {Don't perform length checking for Type field interpretations}
            RemovePad := dataParam
        end
    else if lengthOrTypeParam ≤ maxValidFrame then
        begin
            if lengthOrTypeParam ≤ maxValidFrame then
                begin
                    validLength := {For length interpretations of the Length/Type field, check to determine if value
                                    represented by Length/Type field matches the received clientDataSize};
                    if validLength and not passReceiveFCSType then
                        RemovePad := {Truncate the dataParam (when present) to the value represented by the
                                    lengthOrTypeParam (in octets) and return the result}
                    else RemovePad := dataParam
                end
            end
        end
    end; {RemovePad}

```

ReceiveLinkMgmt attempts repeatedly to receive the bits of a frame, discarding any fragments from collisions by comparing them to the minimum valid frame size:

```

procedure ReceiveLinkMgmt;
begin
    repeat
        StartReceive;
        while receiving do nothing; {Wait for frame to finish arriving}
    until

```

```

    excessBits := frameSize mod 8;
    frameSize := frameSize – excessBits; {TTruncate to octet boundary}
    receiveSucceeding := receiveSucceeding and (frameSize ≥ minFrameSize)
                                                {RReject collision fragments}

    until receiveSucceeding
end; {ReceiveLinkMgmt}

procedure StartReceive;
begin
    receiveSucceeding := true;
    receiving := true
end; {StartReceive}

```

The BitReceiver process runs asynchronously, receiving bits from the medium at the rate determined by the Physical Layer's ReceiveBit operation, partitioning them into frames, and optionally receiving them:

```

process BitReceiver;
    var    b-: PhysicalBit;
           incomingFrameSize: integer; {CCount of all bits received in frame including extension}
           frameFinished: Boolean;
           enableBitReceiver: Boolean;
           currentReceiveBit: 1..frameSize; {Position of current bit in incomingFrame}
begin
    cycle {OOuter loop}
    if receiveEnabled then
        begin {RReceive next frame from physical layer}
            currentReceiveBit := 1;
            incomingFrameSize := 0;
            frameFinished := false;
            enableBitReceiver := receiving;
            PhysicalSignalDecap; {Skip idle and extension, strip off preamble and sfd}
            if enableBitReceiver then extensionOK := true;
            while receiveDataValid and not frameFinished do
                {inner loop to receive the rest of an incoming frame}
                begin {Inner loop to receive the rest of an incoming frame}
                    b := ReceiveBit; {NNext bit from physical medium}
                    incomingFrameSize := incomingFrameSize + 1;
                    if b = 0 or b = 1 then {NNormal case}
                        if enableBitReceiver then {AAppend to frame}
                            begin
                                if incomingFrameSize > currentReceiveBit then extensionOK := false;
                                    {Errors in the extension get mapped to data bits on input}
                                incomingFrame[currentReceiveBit] := b;
                                currentReceiveBit := currentReceiveBit + 1
                            end
                        else if not extending then frameFinished := true; {b must be an extensionBit}
                            if not extending then frameFinished := true;
                        if incomingFrameSize ≥ slotTime then extending := false
                    end; {IInner loop}
                end
            if enableBitReceiver then
                begin
                    frameSize := currentReceiveBit – 1;
                    receiveSucceeding := not extending;
                    receiving := false
                end
            end
        end
    end
end;

```

```

        end
    end {eEnabled}
end {oOuter loop}
end; {BitReceiver}

```

The bits received from the physical layer can take one of three values: data zero (0), data one (1), or extensionBit (EXTEND). The value extensionBit will not occur between the first preamble bit of a frame and the last data bit of a frame in normal circumstances. Extension bits are counted by the BitReceiver but are not appended to the incoming frame. The BitReceiver checks whether the bit received from the physical layer is a data bit or an extensionBit before appending it to the incoming frame. Thus, the array of bits in incomingFrame will only contain data bits. The underlying Reconciliation Sublayer (RS) maps incoming EXTEND_ERROR bits to normal data bits. Thus, the reception of additional data bits after the frame extension has started is an indication that the frame should be discarded.

```

procedure PhysicalSignalDecap;
begin
    {Receive one bit at a time from physical medium until a valid sfd is detected, discard bits and return;}
end; {PhysicalSignalDecap}

```

The process SetExtending controls the extending variable, which determines whether a received frame must be at least slotTime bits in length or merely minFrameSize bits in length to be considered valid by the BitReceiver. SetExtending sets the extending variable to true whenever receiveDataValid is de-asserted, while in half duplex mode ~~at operating speeds above 100 Mb/s, an operating speed of 1000 Mb/s;~~

```

process SetExtending;
begin
    cycle while {HLoop forever}
        while receiveDataValid do nothing;
        extending := extend and halfDuplex
    end {HLoop}
end; {SetExtending}

```

4.2.10 Common procedures

Change 4.2.10 to read as follows:

The function CRC32 is used by both the transmit and receive algorithms to generate a 32-bit CRC value:

```

function CRC32(f: Frame): CRCValue;
begin
    CRC32 := {The 32-bit CRC for the entire frame, excluding the FCS field (if present)}
end; {CRC32}

```

Purely to enhance readability, the following procedure is also defined:

```

procedure nothing; begin end;

```

The idle state of a process (that is, while waiting for some event) is cast as repeated calls on this procedure.

4.3.2 Services provided by the MAC sublayer

Change 4.3.2 to read as follows:

The services provided to the MAC client by the MAC sublayer are transmission and reception of frames. The interface through which the MAC client uses the facilities of the MAC sublayer therefore consists of a pair of functions.

Functions:

TransmitFrame

ReceiveFrame

Each of these functions has the components of a frame as its parameters (input or output), and returns a status code as its result. ~~Note that the service_class defined in 2.3.1 is ignored by the CSMA/CD MAC.~~

NOTE 1—The frame_check_sequence parameter defined in 2.3.1 and 2.3.2 is mapped here into two variables: fcsParamValue and fcsParamPresent. This mapping has been defined for editorial convenience. The fcsParamPresent variable indicates the presence or absence of the fcsParamValue variable in the two function calls. If the fcsParamPresent variable is true, the fcsParamValue variable contains the frame check sequence for the corresponding frame. If the fcsParamPresent variable is false, the fcsParamValue variable is unspecified. If the MAC sublayer does not support client-supplied frame check sequence values, then the fcsParamPresent variable in TransmitFrame shall always be false.

NOTE 2—The mac_service_data_unit parameter defined in 2.3.1 and 2.3.2 is mapped here into two variables: lengthOrTypeParam and dataParam. This mapping has been defined for editorial convenience. The first two octets of the mac_service_data_unit parameter contain the lengthOrTypeParam variable. The remaining octets of the mac_service_data_unit parameter form the dataParam variable.

The MAC client transmits a frame by invoking TransmitFrame:

```
function TransmitFrame(
function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    lengthOrTypeParam: LengthOrTypeValue;
    dataParam: DataValue): TransmitStatus;
    dataParam: DataValue;
    fcsParamValue: CRCValue;
    fcsParamPresent: Bit): TransmitStatus;
```

The TransmitFrame operation is synchronous. Its duration is the entire attempt to transmit the frame; when the operation completes, transmission has either succeeded or failed, as indicated by the resulting status code:

```
type TransmitStatus = (transmitOK, excessiveCollisionError, lateCollisionErrorStatus);
‡ type TransmitStatus = (transmitDisabled, transmitOK, excessiveCollisionError,
    lateCollisionErrorStatus);
```

The transmitDisabled status code indicates that the transmitter is not enabled. Successful transmission is indicated by the status code transmitOK. ~~‡~~ The code excessiveCollisionError indicates that the transmission attempt was aborted due to excessive collisions, because of heavy traffic or a network failure. MACs operating in the half duplex mode at speeds above 100 at the speed of 1000 Mb/s are required to report lateCollisionErrorStatus in response to a late collision; MACs operating in the half duplex mode at speeds of 100 Mb/s and below are not required to do so. TransmitStatus is not used by the service interface defined in 2.3.1. TransmitStatus may be used in an implementation dependent manner.