802.15.5™

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

Part 15.5: Mesh Topology Capability in Wireless Personal Area Networks (WPANs)

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

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

8 May 2009

IEEE Std 802.15.5™-2009
-

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

Part 15.5: Mesh Topology Capability in Wireless Personal Area Networks (WPANs)

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 19 March 2009

IEEE-SA Standards Board

Abstract: This IEEE recommended practice defines the architectural framework that enables WPAN devices to promote interoperable, stable, and scaleable wireless mesh topologies and, if needed, to provide the amendment text to the current WPAN standards that is required to implement this recommended practice.

Keywords: address assignment, block addressing, broadcast, carrier sense multiple access/collision avoidance, high-rate WPAN mesh, HR-WPAN mesh, multicast, low-rate WPAN mesh, LR-WPAN mesh, mesh, mesh coordinator, server routing, wakeup interval, multiple hop, multi-hop, local link state, logical tree, portable, portability, power saving, reliable broadcast, sensor network, traceroute, unicast, wireless PAN, WPAN

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

PDF: ISBN 978-0-7381-5917-1 STD95913
Print: ISBN 978-0-7381-5918-8 STDPD95913

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 or the soundness of any judgments 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 his or her independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

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. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

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

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

Introduction

This introduction is not part of IEEE Std 802.15.5-2009, IEEE Recommended Practice for Information technology—Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements.

This recommended practice provides the architectural framework enabling WPAN devices to promote interoperable, stable, and scalable wireless mesh topologies. This recommended practice is composed of two parts: low-rate WPAN mesh and high-rate WPAN mesh networks. The low-rate mesh is built on IEEE 802.15.4 MAC, while high rate mesh utilizes IEEE 802.15.3/3b MAC. Common features of both meshes include network initialization, addressing, and multihop unicasting. In addition, low-rate mesh supports multicasting, reliable broadcasting, portability support, trace route and energy saving function, and high-rate mesh supports multihop time-guaranteed service.

Notice to users

Laws and regulations

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

Copyrights

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association web site at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA web site at http://standards.ieee.org.

Errata

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

Interpretations

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

Patents

Attention is called to the possibility that implementation of this recommended practice may require use of subject matter covered by patent rights. By publication of this recommended practice, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this recommended practice are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this recommended practice was completed, the IEEE P802.15 Working Group had the following officers:

Dr. Robert F. Heile, *Working Group Chair*
Rick Alfvin, *Working Group Vice Chair*
Patrick Kinney, *Working Group Vice Chair & Secretary*
Dr. James P.K. Gilb, *Working Group Technical Editor*
Dr. John Barr, *Working Group Treasurer*

At the time this recommended practice was completed, the IEEE P802.15 Working Group had the following officers:

Dr. Myung Jong Lee, *Task Group 5 Chair*
Dr. Ho-in Jeon, *Task Group 5 Vice Chair*
Dr. Tae Rim Park, *Task Group 5 Secretary*
Dr. Chunhui (Allan) Zhu, *Task Group 5 Low-Rate Editor*
Dr. Sang Sung Choi, *Task Group 5 High-Rate Editor*

Authur Astrin
Taehan Bae
Jay Bain
Gal Basson
Tuncer Baykas
Phil Beecher
Bruce Bosco
Andre Bourdoux
Pat Carson
Philippe Chambelin
Huor-Hsin Chang
Chang-Soon Choi
Carlos Cordeiro
Alexey Davydov
Hendricus De Ruijter
Paul Dixon
Kai Dombrowski
John Dorsey
Bas Driesen
Amal Ekbal
Yossi Erlich
Robert Fanfelle
John Farserotu
Reed Fisher
Yoshitsugu Fujita
Kiyoshi Fukui
Shigeru Fukunaga
Ryuhei Funada
Uhland Goebel
Paul Gorday
Giriraj Goyal
Eckhard Grass
Mark Grodzinsky
Vivek Gupta
Robert Hall
Christopher J. Hansen
Shinsuke Hara
Hiroshi Harada
Seockdeock Hong
Tian-Wei Huang
Ichirou Ida
Hideto Ikeda
Tetsushi Ikegami
Akio Iso
Beomjin Jeon
Young-Ae Jeon
Seong-Soon Joo
Chol Su Kang
Tae-Gyu Kang
Yasunao Katayama
Shuzo Kato
Yasutaka Kawamoto
Stuart Kerry
Jaehwa Kim
Jae-Hyon Kim
Jinkyeong Kim
Kihong Kim
Kyeongpyo Kim
Seong Kim
Yongsun Kim
Ryota Kimura
Kursat Kimyacioglu
Ryuji Kohno
Fumihide Kojima
Edwin Kwon
Hyoungjin Kwon
Ismail Lakkis
John Lampe
Zhou Lan
Jae Lee
Jeong Lee
Myung Lee
Seong-hee Lee
Taehoon Lee
Wooyong Lee
Zhongding Lei
Daniel Lewis
Huan-Bang Li
Liang Li
Sheung Li
Yong Liu
Alexander Maltsev
Abbie Mathew
Taisuke Matsumoto
Michael Mcinnis
Michael Mclaughlin
Klaus Meyer
Dino Miniutti
Rajendra Moorti
Jorge Myszne
Yukimasa Nagai
Ken Naganuma
Chiu Ngo
Paul Nikolich
Yoshinori Nishiguchi
Hiroyo Ogawa
Jisung Oh
Laurent Ouvry
Pascal Pagani
Tae Rim Park
Maulin Patel
Stephane Pinel
Frank Poegel
Stephen Pope
Clinton Powell
Chang Woo Pyo
Xiangping Qin
Ivan Reede
Richard Roberts
Benjamin A. Rolfe
Ali Sadri
Katsuyoshi Sato
Hirokazu Sawada
Kamran Sayrafian
Michael Schmidt
Jean Schwoerer
Huai-Rong Shao
Stephen Shellhammer
Shusaku Shimada
Yukihiro Shimakata
Chang Sub Shin
Michael Sim
Harkirat Singh
Carl Stevenson
Paul Strauch
Chin Sum
Kazuaki Takahashi
Kenichi Takizawa
Arnaud Tonnerre
Ichihiko Toyoda
Jason Trachewsky
Solomon Trainin
Alberto Valdes Garcia
Magnus Wiklund
Gerald Wineinger
Ludwig Winkel
Eun Tae Won
Jongeun Won
Pengfei Xia
Kamya Yazdandoost
James Yee
Kaoru Yokoo
Su Yong
Zhan Yu
Bin Zhen
Chunhui Zhu

Major contributions for this recommended practice were received from the following individuals:

James Allen
Jay Bain
Phil Beecher
John Boot
Ishikawa Chiaki
Sungrae Cho
Sang Sung Choi
Wonsuk Choi
Chun-Ting Chou
Francis daCosta
Klaus Fosmark
James P.K. Gilb
Uhland Goebel
Guido R. Hiertz
Ho-In Jeon
Young-Ae Jeon
Seong-Soon Joo
Hakyung Jung
Patrick Kinney
Seongjae Kwon
Inhwan Lee
Myung Jong Lee
Seong-hee Lee
Yong Liu
Sebastian Max
Klaus Meyer
Laihyuk Park
Sung-Woo Park
Frank Poegel
Hans-Jürgen Reumerman
Seung Hyong Rhee
Benjamin A. Rolfe
Michael Schmidt
Huai-Rong Shao
Chang Sub Shin
Michael Sim
Marcus Wong
June Yoon
Rui Zhang
Jianliang Zheng
Chunhui Zhu

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

Richard Alfvin
James Allen
Danilo Antonelli
Philip Beecher
Lyle Bullock
William Byrd
James Carlo
Sang Sung Choi
Keith Chow
Charles Cook
Todor Cooklev
Thomas Dineen
Sourav Dutta
Paul Eastman
Richard Eckard
Marc Emmelmann
Avraham Freedman
Devon Gayle
James P.K. Gilb
Randall Groves
C. Guy
Rainer Hach
John Hawkins
Marco Hernandez
Atsushi Ito
Beomjin Jeon
Young-Ae Jeon
Bobby Jose
Kaku, Shinkyo
Stuart J. Kerry
Yongbum Kim
Patrick Kinney
Cees Klik
Joseph Kubler
Myung Jong Lee
Seong-hee Lee
Jan-Ray Liao
Arthur Light
William Lumpkins
Peter Martini
Michael Mcinnis
Wade Midkiff
Avygdor Moise
Hiroyuki Nakase
Michael S. Newman
Charles Ngethe
John Notor
Satoshi Obara
Knut Odman
Okundu Omeni
Satoshi Oyam
Tae Rim Park
Eldad Perahia
Subburajan Ponnuswamy
Clinton Powell
Robert Robinson
Benjamin Rolfe
Shigenobu Sasaki
Peter Saunderson
Bartien Sayogo
Chang Sub Shin
Kapil Sood
Amjad Soomro
Thomas Starai
Rene Struik
Walter Struppler
Mark Sturza
Thomas Tullia
Dmitri Varsanofiev
Prabodh Varshney
Stanley Wang
Andreas Wolf
Paul Work
Oren Yuen
Wenhao Zhu
Juan Zuniga

When the IEEE-SA Standards Board approved this recommended practice on 19 March 2009, it had the following membership:

Robert M. Grow, *Chair*
Thomas Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

John Barr
Karen Bartleson
Victor Berman
Ted Burse
Dick DeBlasio
Andy Drozd
Mark Epstein
Alexander Gelman
Jim Hughes
Richard Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Ted Olsen
Glenn Parsons
Ron Petersen
Narayanan Ramachandran
Jon Walter Rosdahl
Sam Sciacca
Howard Wolfman

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Michael Janezic, *NIST Representative*

Michelle D. Turner
IEEE Standards Program Manager, Document Development

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

Contents

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

Part 15.5: Mesh Topology Capability in Wireless Personal Area Networks (WPANs)

1. Overview

A Wireless Personal Area Network (WPAN) mesh is a network of WPANs with mesh topology. This document defines a recommended practice for WPAN meshes.

1.1 Scope

The scope of this standard is to provide a recommended practice to provide the architectural framework enabling WPAN devices to promote interoperable, stable, and scaleable wireless mesh topologies and, if needed, to provide the amendment text to the current WPAN standards that is required to implement this recommended practice.

1.2 Purpose

The purpose of this project is to facilitate wireless mesh topologies optimized for IEEE 802.15 WPANs.

Mesh Topology provides the following features to WPANs:

— Extension of network coverage without increasing the transmit power or the receiver sensitivity

— Enhanced reliability via route redundancy

— Easier network configuration

— Better device battery life

2. Normative references

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

IEEE Std 802.15.3™-2003, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs). [1, 2]

IEEE Std 802.15.3b™-2005, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs) Amendment 1: MAC Sublayer.

IEEE Std 802.15.4™-2006, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs).

3. Definitions

For the purposes of this draft recommended practice, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms* should be referenced for terms not defined in this clause.

3.1 active duration: A time period defined in both synchronous and asynchronous energy saving modes in the mesh sublayer during which a mesh device accesses the common channel using carrier sense multiple access with collision avoidance mechanism.

[1] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[2] The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

3.2 ancestor: A device that is closer to the root of the logical tree than this device along the tree.

3.3 child device: A device that associates with the device. A child device's tree level is one more than that of the device. If a device's tree level equals to i, then all of its child devices (children) will have tree level equal to i+1.

3.4 descendant: A device that is farther from the root of the logical tree than the device.

3.5 end device: A device that is not capable of relaying data frames for other devices.

3.6 flooding: A broadcast method in which each device, other than the original sender, receives and relays the same broadcast data frames exactly once.

3.7 group coordinator: The controller of a specific multicast group.

3.8 group member: A device participates in a multicast communication.

3.9 inactive duration: A time period defined in both synchronous and asynchronous energy saving modes in mesh sublayer during which a mesh device may turn off its receiver circuitry to save energy. An inactive duration follows an active duration in time.

3.10 extended neighbor: A device less than or equal to meshExtendedNeighborHopDistance-hops away from the device, where meshExtendedNeighborHopDistance is a Mesh Information Base (MeshIB) attribute. Neighbors more than one hop away are not able to talk to each other directly.

3.11mesh address: A logical address assigned to a mesh device in the mesh sublayer. Each mesh address is unique in a mesh network so that a mesh device can be uniquely identified.

3.12 mesh coordinator: An entity that manages the mesh network. It refers to the root of the logical tree of a mesh network whose tree level equals zero.

3.13 mesh device: A device that is capable of relaying data frames for other devices.

3.14 mesh discovery table: A temporary table that records the information of all neighbor devices discovered by the mesh discovery process.

3.15 mesh sublayer timer: A timer for an optional energy saving mode for mesh sublayer.

3.16 mobility: A feature that allows a device to be moved from one location to another within the network coverage while maintaining the communication session during the move.

3.17 multicast router: A device that is not a multicast group member but relays multicast data frames for one or more multicast groups.

3.18 multicast agent: A device with at least one of its end devices that is a member of a multicast group.

3.19 neighbor: A device that is in the communications range of the current device.

3.20 neighbor MPNC: MPNCs that are single hop away from an MPNC with a direct link.

3.21 optimal route: A minimal cost route which is provided by topology server.

3.22 portability: A feature that allows a device to be moved from one location to another within the network coverage and to resume operation at the new location. In contrast to mobility, the communication session is not maintained during the move.

3.23 region synchronizer: A mesh device that is responsible for sending synchronization request frames to its child devices.

3.24 residual CTA time: The channel time assignment time available at the MPNC.

3.25 server routing: A routing method which is done by topology servers distributed over the mesh network.

3.26 sibling device: A device's one-hop neighbor that is neither its parent nor its children.

3.27 synchronization duration: The time duration of the synchronization procedure in the optional synchronous power saving mode.

3.28 synchronization interval: The time between two consecutive synchronization processes.

3.29 synchronization region: The hop distance to be synchronized by a mesh coordinator or region synchronizer in the optional synchronous power saving mode.

3.30 sync-child: A logical child device that is responsible for maintaining synchronization with its sync-parent device.

3.31 sync-parent: A logical parent device that is responsible for maintaining synchronization with its sync-child devices.

3.32 tree level: The distance, in the number of hops, between a device and the root along the logical tree. The root of the tree always has a tree level of zero.

3.33 wakeup interval: The time period defined for the optional energy saving mode for mesh sublayer. It consists of an active duration and an inactive duration.

4. Acronyms

ABAT	adaptive block addressing table
AD	active duration
ASES	asynchronous energy saving
AO	active order
APDU	application protocol data unit
BAT	block addressing table
BcstID	broadcast identifier
BSID	beacon source identifier
CTA	channel time allocation
DestID	destination identifier
DA	destination address

DEV	device
DEVID	device identifier
Dly_ACK	delayded acknowledgment
DME	device management entity
DO	destination active order
EDEV	end device
ARC	asynchronous energy saving retry counter
EREP	extension reply
EREQ	extension request
FC	frame control
FCSL	frame convergence sublayer
GC	group coordinator
G-JREQ	group join request
G-JREP	group join reply
GM	group member
HDTV	high definition TV
IE	information element
IMM_ACK	immediate acknowledgment
LLC	logical link control
LST	link state table
LQI	link quality indication
MAC	medium access control
MC	mesh coordinator
MDEV	mesh-capable device
MeshID	mesh network identifier
MHME	mesh sublayer management entity
MHPDU	mesh protocol data unit
MHSDU	mesh service data unit

MeshIB	mesh information base
MPNC	mesh capable piconet coordinator
MPNCID	mesh-capable piconet coordinator identifier
MSC	message sequence chart
MTR	multicast transaction record
MTT	multicast transaction table
MDT	mesh discovery table
OUI	organizationally unique identifier
PDU	protocol data unit
PHY	physical layer
PIB	PAN information base
PLME	PHY layer management entity
PN	piconet
PNC	piconet coordinator
PNCID	piconet coordinator identifier
PNID	piconet identifier
RT	router
SA	source address
SAP	service access point
SES	synchronous energy saving
TREEID	tree network identifier
TTL	time to live
WI	wakeup interval
WN	wakeup notification
WO	wakeup order
WPAN	wireless personal area network

5. Low-rate WPAN mesh

5.1 General description

This clause of the recommended practice provides an architectural framework that enables low-power, low-rate WPAN devices to promote interoperable, stable, and scalable wireless mesh topologies. Mesh network topologies, as illustrated in Figure 1, allow devices with IEEE Std 802.15.4-2006[3] compatible MAC/PHY to extend the network coverage without increasing the transmission power or the receiver sensitivity. Another key advantage of the mesh network is the enhanced reliability via route redundancy. Mesh networking topology is one of the two topologies IEEE Std. 802.15.4-2006 was designed to support, as described in 5.3 of IEEE Std 802.15.4-2006. However, the tasks of defining and implementing the mesh networking functions are out of the scope of IEEE Std 802.15.4-2006. This recommended practice provides a standard way of doing mesh networking over IEEE Std 802.15.4-2006 within the IEEE standard body.

Supported features for LR-WPAN include the following:

— unicast, multicast, and reliable broadcast mesh data forwarding

— synchronous and asynchronous power saving for mesh devices

— trace route function

— portability of end devices

End-to-end reliability and security are not provided in the mesh sublayer in this version of the recommended practice.

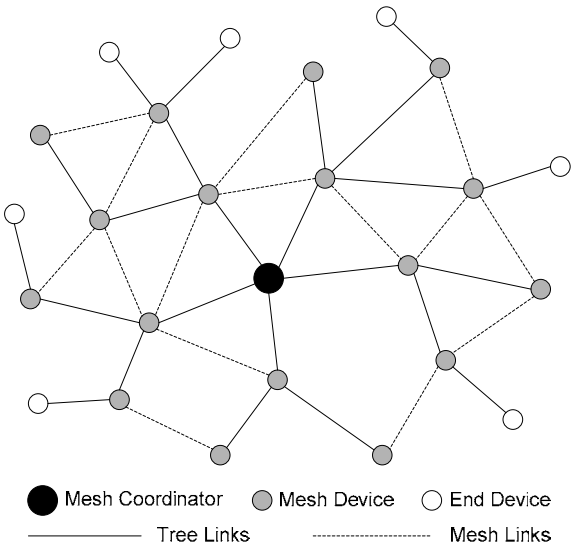

Figure 1—An illustration of mesh network topology

The applications of low rate WPAN mesh networks are numerous; including but not limited to, home and industrial automation and control, security and environmental monitoring, situational awareness and asset

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

tracking, automatic meter reading, personal health monitoring, and etc. Example use cases could be as follows:

a) A RF remote control that can control all AV devices in all rooms of a residential house without line of sight requirement

b) Wireless light switches in a commercial building, e.g. a department store, control lights of the entire floor, with the capability of grouping lights in different ways in a dynamic manner and turn them on/off with one push of the button

c) Smart utility meters form a mesh network, collect meter reading data and send the data to utility companies through gateways

In Clause 5 of this document, the terms *mesh network* and *PAN* are used alternatively to refer to the mesh network since all devices in a mesh network share the same PAN ID. In other words, inter-PAN communication is not supported in the current version of the standard.

The reference model of low rate WPAN mesh is illustrated in Figure 2.

In the data plane, the WPAN Mesh Sublayer resides between the Service Specific Convergence Sublayer (SSCS) of IEEE Std 802.2-1998 Logical Link Control (LLC) and the IEEE Std 802.15.4-2006 Media Access Control (MAC) Sublayer. Mesh Sublayer provides services to the next higher layer via Mesh Service Access Point (Mesh SAP). For implementers to add mesh function to the existing low rate WPAN applications with the least effort, the Mesh SAP is made almost identical to the MCPS SAP.

In the management plane, the mesh sublayer management entity (MHME) resides between the application management entity (AME) and the MAC layer management entity (MLME). MHME also interfaces with the mesh sublayer at the same level. As a reference, the device management entity (DME) that has access and control to all layers is also shown in the figure. However, the specifications of both AME and DME are beyond the scope of this document.

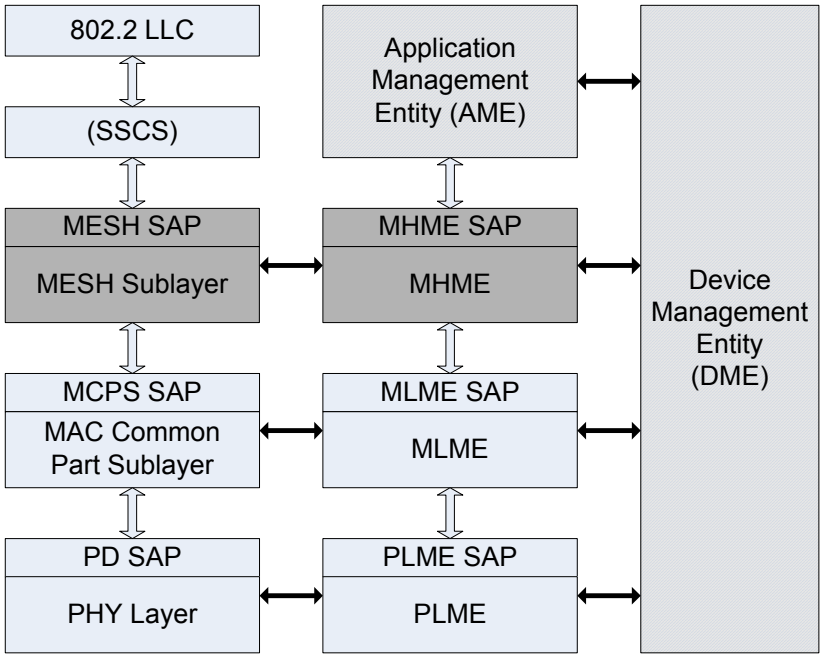

Figure 2—The reference model for low-rate WPAN mesh

5.2 Mesh service description

The mesh sublayer provides two types of services, the mesh data service and the mesh management service, to the next higher layer via two corresponding service access points, the mesh SAP and MHME SAP. There is also an internal interface between the mesh sublayer and the MHME allowing MHME to utilize the mesh sublayer data service. On the other hand, the mesh sublayer may also need to get information from MHME through this internal interface to compose mesh sublayer data frames.

5.2.1 Mesh data service

The function of the mesh sublayer data service is to support the transport of application protocol data units (APDUs) between peer application entities residing at different devices that can be multiple hops away from each other. As described in 5.1, the mesh data service primitives are very similar to the MCPS SAP primitives. The implementers should pay careful attention to the differences between them. Table 1 lists the primitives supported by the mesh SAP.

Table 1—Mesh SAP primitives

MESH SAP primitives	Request	Confirm	Indication
MESH-DATA	5.2.1.1	5.2.1.2	5.2.1.3
MESH-PURGE	5.2.1.4	5.2.1.5	

5.2.1.1 MESH-DATA.request

The MESH-DATA.request primitive requests the transfer of an SPDU (SSCS protocol data unit), from a local SSCS entity to one or more peer SSCS entities. It is very similar to the MCPS-DATA.request primitive at MAC layer.

The semantics of the MESH-DATA.request primitive is as follows:

```
MESH-DATA.request    (
                     SrcAddrMode,
                     DstAddrMode,
                     DstAddr,
                     mhsduLength,
                     mhsdu,
                     mhsduHandle,
                     AckTransmission,
                     McstTransmission,
                     BcstTransimission,
                     ReliableBcst
                     )
```

Table 2 specifies the parameters for the MESH-DATA.request primitive.

Table 2—MESH-DATA.request parameters

Name	Type	Valid range	Description
SrcAddrMode	Integer	0x00–0x03	The source addressing mode for this primitive and subsequent MHPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstAddrMode	Integer	0x00–0x03	The destination addressing mode for this primitive and subsequent MHPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstAddr	Device address	As specified by the DstAddrMode parameter	The device address of the entity, or entities in the case of multicast and broadcast, to which the MHSDU is being transferred.
mhsduLength	Integer	≤ *(aMaxMACSafePayloadSize - meshcMaxMeshHeaderLength)*	The number of octets contained in the MHSDU to be transmitted by the mesh sublayer entity.
mhsdu	Set of octets	–	The set of octets forming the MHSDU to be transmitted by the mesh sublayer entity.
mhsduHandle	Integer	0x00–0xff	The handle associated with the MHSDU to be transmitted by the mesh sublayer entity.
AckTransmission	Boolean	TRUE, FALSE	This field is set to TRUE if an acknowledgement is required from the receiver; otherwise, it is set to FALSE.
McstTransmission	Boolean	TRUE, FALSE	This field is set to TRUE if the data is to be multicast; otherwise, it is set to FALSE.
BcstTransimission	Boolean	TRUE, FALSE	This field is set to TRUE if the data is to be broadcast; otherwise, it is set to FALSE.
ReliableBcst	Boolean	TRUE, FALSE	This field is set to TRUE if reliable broadcast is required; otherwise, it is set to FALSE. It is meaningful only when BcstTransmission is set to TRUE.

The MESH-DATA.request primitive is generated by a local SSCS entity when an SPDU is to be transferred to a peer SSCS entity.

On receipt of the MESH-DATA.request primitive, the mesh sublayer of a device first looks at the value of the DstAddrMode field. If the value indicates a 64-bit extended address is used (0x03), the data may be sent to its one-hop neighbors only. Therefore, the mesh sublayer should compose an MCPS-DATA.request and send it to the MAC layer. In this case, the mesh sublayer simply passes the request from the higher layer without calling the mesh sublayer functions. b_0 of the TxOptions field of the MCPS-DATA.request primitive should reflect the value of AckTransmission; it is set to one when the AckTransmission field is set to TRUE and set to zero when the AckTransmission field is set to FALSE. Fields b_1 and b_2 of the TxOptions of the MCPS-DATA.request primitive should always be set to zero. The rest of the transmit option fields, McstTransmission, BcstTransmission, and ReliableBcst, are created to indicate to the mesh sublayer the transmission mode of the data. Therefore they are only meaningful to the mesh sublayer and should not be reflected in the MCPS-DATA.request primitive.

If the value of the DstAddrMode field indicates a 16-bit short address is used (0x02) the destination could be either a one-hop neighbor or a device multiple hops away. Therefore, the device should first determine whether the destination is one of its one-hop neighbors by looking at the DstAddr and searching the destination address in its neighbor list as illustrated in Table 46. If it is, the mesh sublayer should compose an MCPS-DATA.request and send it to the MAC layer as described above.

If the destination is not a one-hop neighbor, the mesh sublayer should determine the next hop in the mesh network to route the data message to its final destination. The mesh sublayer should first compose an MHPDU to transmit the SPDU received from SSCS as described in 5.3.2.1. The mesh sublayer should then look up the next hop toward the destination using the method described in 5.5.5 and compose an MCPS-DATA.request with the MAC layer destination address set to the address of the next hop. The MCPS-DATA.request primitive is then sent to the MAC layer.

Because the MHSDU has now been wrapped by the mesh header, the payload to be sent to the MAC layer will be the MHPDU. The mhsduLength should be equal to or less than *(aMaxMACFrameSize – meshcMaxMeshHeaderLength),* where *meshcMaxMeshHeaderLength* is a mesh constant in Table 41. The values in fields other than those discussed above should be kept unchanged and pass through to the MAC layer.

If the destination is not an one-hop neighbor, the values of all four transmit option fields should be recorded in the corresponding bits of the Transmission Mode field of the Frame Control field in the composed mesh sublayer data frame as illustrated in Figure 5 and transmitted to the next hops so that the transmission mode information can be carried while the data is propagated. Note values of the transmit option fields only have local meanings. For example, when AckTransmission is set to TRUE to indicate an acknowledged transmission, it only means the MAC layer of this device should transmit the data frame using acknowledged transmission and does not mean end-to-end acknowledgement is supported.

If the MHPDU was successfully transmitted and an acknowledgment, if requested, was received, the mesh sublayer will issue the MESH-DATA.confirm primitive with a status of SUCCESS.

If any parameter in the MESH-DATA.request primitive is not supported or is out of range, the mesh sublayer will issue the MESH-DATA.confirm primitive with a status of INVALID_PARAMETER.

5.2.1.2 MESH-DATA.confirm

The MESH-DATA.confirm primitive reports the results of a request to transfer a data SPDU (MHSDU) from a local SSCS entity to one or more peer SSCS entities.

The semantics of the MESH-DATA.confirm primitive is as follows:

```
MESH-DATA.confirm    (
                     mhsduHandle,
                     Status
                     )
```

Table 3 specifies the parameters for the MESH-DATA.confirm primitive.

Table 3—MESH-DATA.confirm parameters

Name	Type	Valid range	Description
mhsduHandle	Integer	0x00–0xff	The handle associated with the MHSDU being confirmed.
Status	Enumeration	SUCCESS, TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, NO_ACK, UNAVAILABLE_KEY, FRAME_TOO_LONG, INVALID_PARAMETER	The status of the last MHSDU transmission.

The MESH-DATA.confirm primitive is generated by the mesh sublayer in response to an MESH-DATA.request primitive. The MESH-DATA.confirm primitive returns a status of either SUCCESS, indicating that the request to transmit was successful, or an error code of TRANSACTION_OVERFLOW, TRANSACTION_EXPIRED, CHANNEL_ACCESS_FAILURE, UNAVAILABLE_KEY, NO_ACK, FRAME_TOO_LONG, or INVALID_PARAMETER. The reasons for these status values are fully described in 7.1.1.1.3 of IEEE Std 802.15.4-2006.

On receipt of the MESH-DATA.confirm primitive, the SSCS of the initiating device is notified of the result of its request to transmit. If the transmission attempt was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter will indicate the error.

5.2.1.3 MESH-DATA.indication

The MESH-DATA.indication primitive indicates the transfer of a data SPDU (i.e., MHSDU) from the mesh sublayer to the local SSCS entity.
The semantics of the MESH-DATA.indication primitive is as follows:

```
MESH-DATA.indication                    (
                                        SrcAddrMode,
                                        SrcPANId,
                                        SrcAddr,
                                        mhsduLength,
                                        mhsdu
                                        )
```

Table 4 specifies the parameters for the MESH-DATA.indication primitive.

The MESH-DATA.indication primitive is generated by the mesh sublayer and issued to the SSCS upon receiving the corresponding MCPS-DATA.indication from the MAC layer that passes the appropriate message filtering operations.

On receipt of the MESH-DATA.indication primitive, the SSCS is notified of the arrival of data at the device.

Table 4—MESH-DATA.indication parameters

Name	Type	Valid range	Description
SrcAddrMode	Integer	0x00–0x03	The source addressing mode for this primitive corresponding to the received MHPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
SrcPANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the entity from which the MHSDU was received.
SrcAddr	Device address	As specified by the SrcAddrMode parameter	The individual device address of the entity from which the MHSDU was received.
mhsduLength	Integer	≤ *aMaxMACFrameSize*	The number of octets contained in the MHSDU being indicated by the MESH sublayer entity.
mhsdu	Set of octets	–	The set of octets forming the MHSDU being indicated by the MESH sublayer entity.

5.2.1.4 MESH-PURGE.request

The MESH-PURGE.request primitive allows the next higher layer to purge an MHSDU from the transaction queue.

The semantics of the MESH-PURGE.request primitive is as follows:

```
MESH-PURGE.request (
                   mhsduHandle
                   )
```

Table 5 specifies the parameters for the MESH-PURGE.request primitive.

Table 5—MESH-PURGE.request parameters

Name	Type	Valid range	Description
mhsduHandle	Integer	0x00–0xff	The handle of the MHSDU to be purged from the transaction queue.

The MESH-PURGE.request primitive is generated by the next higher layer whenever an MHSDU is to be purged from the transaction queue.

On receipt of the MESH-PURGE.request primitive, the mesh sublayer attempts to find in its transaction queue the MHSDU indicated by the mhsduHandle parameter. If an MHSDU matching the given handle is found, the MHSDU is discarded from the transaction queue, and the mesh sublayer issues the MESH-PURGE.confirm primitive with a status of SUCCESS. If an MHSDU matching the given handle is not found, the mesh sublayer issues the MESH-PURGE.confirm primitive with a status of INVALID_HANDLE.

5.2.1.5 MESH-PURGE.confirm

The MESH-PURGE.confirm primitive allows the mesh sublayer to notify the next higher layer of the success of its request to purge an MHSDU from the transaction queue.

The semantics of the MESH-PURGE.confirm primitive is as follows:

```
MESH-PURGE.confirm          (
                              mhsduHandle,
                              Status
                            )
```

Table 6 specifies the parameters for the MESH-PURGE.confirm primitive.

Table 6—MESH-PURGE.confirm parameters

Name	Type	Valid range	Description
mhsduHandle	Integer	0x00–0xff	The handle of the MHSDU requested to be purged from the transaction queue.
Status	Enumeration	SUCCESS or INVALID_HANDLE	The status of the request to purge an MHSDU from the transaction queue.

The MESH-PURGE.confirm primitive is generated by the mesh sublayer in response to an MESH-PURGE.request primitive. The MESH-PURGE.confirm primitive returns a status of either SUCCESS, indicating that the purge request was successful, or INVALID_HANDLE, indicating an error. The reasons for these status values are fully described in 7.1.1.4.3 of IEEE Std 802.15.4-2006.

On receipt of the MESH-PURGE.confirm primitive, the next higher layer is notified of the result of its request to purge an MHSDU from the transaction queue. If the purge request was successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter will indicate the error.

5.2.1.6 Data service message sequence chart

Figure 3 illustrates the sequence of messages necessary for a successful data transfer between two devices.

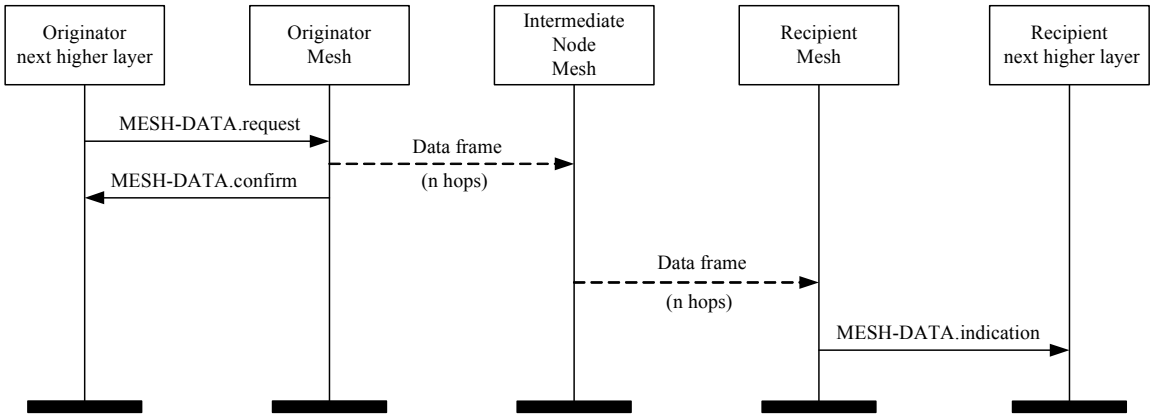

Figure 3—Message sequence chart describing the mesh data service

5.2.2 Mesh management service

The MHME-SAP allows the transport of management commands between SSCS and the MHME Table 7 summarizes the primitives supported by the MHME through the MHME-SAP interface. The primitives are discussed in the subclauses referenced in the table.

Table 7—Summary of the primitives accessed through the MHME-SAP

Name	Subclause Number in This Recommended Practice				Mandotary/ Optional
	Request	Indication	Response	Confirm	
MHME-DISCOVER	5.2.2.1			5.2.2.2	M
MHME-START-NETWORK	5.2.2.3			5.2.2.4	M
MHME-START-DEVICE	5.2.2.5			5.2.2.6	M
MHME-JOIN	5.2.2.7	5.2.2.8		5.2.2.9	M
MHME-LEAVE	5.2.2.10	5.2.2.11		5.2.2.12	M
MHME-RESET	5.2.2.13			5.2.2.14	M
MHME-GET	5.2.2.15			5.2.2.16	M
MHME-SET	5.2.2.17			5.2.2.18	M
MHME-START-SYNC	5.2.2.19			5.2.2.20	O

Table 7— Summary of the primitives accessed through the MHME-SAP *(continued)*

Name	Subclause Number in This Recommended Practice				Mandotary/ Optional
	Request	Indication	Response	Confirm	
MHME-TRACE-ROUTE	5.2.2.21	5.2.2.22		5.2.2.23	O
MHME-MULTICAST-JOIN	5.2.2.24			5.2.2.25	M
MHME-MULTICAST-LEAVE	5.2.2.26			5.2.2.27	M

5.2.2.1 MHME-DISCOVER.request

This primitive allows the next higher layer to request the mesh sublayer to discover mesh networks currently operating within the neighborhood.

The semantics of the primitive is as follows:

MHME-DISCOVER.request (
ScanChannels,
ScanDuration,
ChannelPage,
ReportCriteria
)

Table 8 specifies the parameters of the MHME-DISCOVER.request primitive.

Table 8—MHME-DISCOVER.request parameters

Name	Type	Valid range	Description
ScanChannels	Bit field	27-bit field	The 27 bits (b_0, b_1,... b_{26}) indicate which channels are to be scanned (1 = scan, 0 = do not scan) for each of the 27 channels supported by the ChannelPage parameter as defined in IEEE Std 802.15.4-2006.
ScanDuration	Integer	0–14	A value used to calculate the length of time to spend scanning each channel: The time spent scanning each channel is (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter. For more information on MAC sublayer scanning as defined in IEEE Std 802.15.4-2006.
ChannelPage	Integer	0–31	The channel page on which to perform the scan.

Table 8— MHME-DISCOVER.request parameters *(continued)*

Name	Type	Valid range	Description
ReportCriteria	Enumeration	LINK_QUALITY, TREE_LEVEL	The field indicates which criterion is used to select the best neighbor to be reported to the next higher layer. LINK_QUALITY = The neighbor with highest link quality is reported. TREE_LEVEL = The neighbor with lowest tree level is reported. Combination of multiple criteria is allowed despite the algorithm is out of scope of this recommended practice.

This primitive is generated by the next higher layer and issued to the mesh sublayer to request the discovery of networks operating within the device's operating area.

On receipt of this primitive, the mesh sublayer will attempt to discover networks operating within the device's operating area by performing an active scan over the channels specified in the ScanChannels argument, and the channel pages specified in the ChannelPage argument, for the period specified in the ScanDuration parameter. The scan is performed by means of the MLME-SCAN.request primitive.

On receipt of the MLME-SCAN.confirm primitive, a device should first record the neighbor devices from which it received the signals in the mesh discovery table (MDT), if the PANDescriptorList of the MLME-SCAN.confirm is not empty. Each beacon found during a scan will create an entry, PANDescriptor, in the PANDescriptorList at the MAC layer and a corresponding entry, MeshDescriptor, in the MDT at the mesh sublayer. The elements of a MeshDescriptor are not exactly the same as that of a PANDescriptor. Elements of MeshDescriptor are illustrated in Table 9. Note the MDT is a temporary table and its content can be cleared after the device has successfully started or joined a mesh network. The mesh sublayer information in beacon payload is defined in Figure 37.

Table 9—Elements of MESHDesriptor

Name	Type	Valid range	Description
PANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the network discovered.
Address	Integer	0x0000–0xfffd	The 16-bit short address of the beacon sender.
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY as defined in IEEE Std 802.15.4-2006.	The current logical channel occupied by the network.
ChannelPage	Integer	Selected from the available channel pages supported by the PHY.	The current channel page occupied by the network.

Table 9— Elements of MESHDesriptor *(continued)*

Name	Type	Valid range	Description
MeshVersion	Integer	0x00–0x0f	The version of the mesh sublayer protocol in use in the discovered network.
BeaconOrder	Integer	0x0f	This specifies how often the MAC sublayer beacon is to be transmitted by a given device on the network. For this version of the recommended practice, the value is always set to 0x0f indicating no periodic beacons are transmitted. For a discussion of MAC sublayer beacon order, please refer to IEEE Std 802.15.4-2006.
SuperframeOrder	Integer	0x0f	For beacon-oriented networks, that is, beacon order < 15, this specifies the length of the active period of the superframe. For this version of the recommended practice, the value is always set to 0x0f indicating no periodic beacons are transmitted. For a discussion of MAC sublayer superframe order, please refer to IEEE Std 802.15.4-2006.
LinkQuality	Integer	0x00–0xff	The LQI value at which the network beacon was received. Lower values represent lower LQI.
TreeLevel	Integer	0x00–0xff	The tree level of the beacon sender.
AcceptMeshDevice	Boolean	TRUE or FALSE	This field indicates whether the mesh network accepts new mesh devices. A value of TRUE indicates the device is capable of accepting join requests from other mesh devices; the value should be set to FALSE otherwise.
AcceptEndDevice	Boolean	TRUE or FALSE	This field indicates whether the mesh network accepts new end devices. A value of TRUE indicates the device is capable of accepting join requests from end devices; the value should be set to FALSE otherwise.
SyncEnergySaving	Boolean	TRUE or FALSE	This field indicates whether the optional synchronous energy saving feature is supported. A value of TRUE indicates the device is capable of supporting synchronous energy saving; the value should be set to FALSE otherwise.

Table 9— Elements of MESHDesriptor *(continued)*

Name	Type	Valid range	Description
AsyncEnergySaving	Boolean	TRUE or FALSE	This field indicates whether the optional asynchronous energy saving feature is supported. A value of TRUE indicates the device is capable of supporting asynchronous energy saving; the value should be set to FALSE otherwise.

The mesh sublayer should then issue the MHME-DISCOVER.confirm primitive containing the information about the discovered networks with a Status parameter value equal to that returned with the MLME-SCAN.confirm.

5.2.2.2 MHME-DISCOVER.confirm

This primitive reports to its next higher layer the result of a network discovery operation.

The semantics of the primitive is as follows:

```
MHME-DISCOVER.confirm    (
                         Status,
                         NetworkCount,
                         MeshDescriptorList
                         )
```

Table 10 specifies the parameters of the MHME-DISCOVER.confirm primitive.

Table 10—MHME-DISCOVER.confirm parameters

Name	Type	Valid range	Description
Status	Status	Any Status value returned with the MLME-SCAN.confirm primitive	Defined in 7.1.11.2 of IEEE Std 802.15.4-2006.
NetworkCount	Integer	0x00–0xff	Gives the number of networks discovered by the search.
MeshDescriptorList	List of mesh network descriptors	The list contains the number of elements given by the NetworkCount parameter	A list of descriptors, one for each of the mesh network discovered.

This primitive is generated by the MHME and issued to its next higher layer on completion of the discovery task initiated by an MHME-DISCOVERY.request primitive.

Table 9 gives a detailed account of the contents of a MeshDescriptor from the MeshDescriptorList parameter. However, not all the entries in the MDT are reported to the next higher layer through the MHME-DISCOVER.confirm primitive; among the beacon senders with the same PANId, only the best one with regard to the ReportCriteria should be reported to the next higher layer. Determining the report criteria is an implementation issue. This standard suggests consider either LinkQuality or TreeLevel or the combination of the two of the discovered devices.

On receipt of this primitive, the next higher layer is notified of the results of a network search.

5.2.2.3 MHME-START-NETWORK.request

This primitive allows the next higher layer to request that the device start a new mesh network with itself as the MC.

The semantics of the primitive is as follows:

```
MHME-START-NETWORK.request      (
                                ScanChannels,
                                ScanDuration,
                                ChannelPage,
                                BeaconOrder,
                                SuperframeOrder
                                )
```

Table 11 specifies the parameters for the MHME-START-NETWORK.request primitive.

Table 11—MHME-START-NETWORK.request parameters

Name	Type	Valid range	Description
ScanChannels	Bit field	27-bit field	The 27 bits (b_0, b_1,... b_{26}) indicate which channels are to be scanned (1 = scan, 0 = do not scan) for each of the 27 channels supported by the ChannelPage parameter as defined in IEEE Std 802.15.4-2006.
ScanDuration	Integer	0–14	A value used to calculate the length of time to spend scanning each channel: The time spent scanning each channel is (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter. For more information on MAC sublayer scanning, please refer to 7.1.11.1 of IEEE Std 802.15.4-2006.
ChannelPage	Integer	0–31	The channel page on which to perform the scan.
BeaconOrder	Integer	0x0f	The beacon order of the network that the higher layers wish to form. For this version of the recommended practice, the value is always set to 0x0f indicating no periodic beacons are transmitted.
SuperframeOrder	Integer	0x0f	The superframe order of the network that the higher layers wish to form. For this version of the recommended practice, the value is always set to 0x0f indicating no periodic beacons are transmitted.

This primitive is generated by the next higher layer of a mesh coordinator-capable device and issued to its MHME to request the initialization of itself as the coordinator of a new network.

On receipt of this primitive by a device that is not capable of being a mesh coordinator (MC) (by checking the mesh sublayer constant, *meshcCoordinatorCapability*), the MHME issues the MHME-START-NETWORK.confirm primitive with the Status parameter set to INVALID_REQUEST.

If the device is to be initialized as an MC, the MHME requests that the MAC sublayer first perform an energy detection scan and then an active scan on the specified set of channels. To do this, the MHME issues the MLME-SCAN.request primitive to the MAC sublayer with the ScanType parameter set to indicate an energy detection scan to find a clean channel and then issues the primitive again with the ScanType parameter set to indicate an active scan. After the completion of the active scan, on receipt of the MLME-SCAN.confirm primitive from the MAC sublayer, the MHME selects a suitable channel. The mesh sublayer will pick a PAN identifier that does not conflict with that of any network known to be operating on the chosen channel. Once a suitable channel and PAN identifier are found, the MHME will choose 0x0000 as the 16-bit short MAC address and inform the MAC sublayer. To do this, the MHME issues the MLME-SET.request primitive to the MAC sublayer to set the MAC PIB attribute *macShortAddress*. At the same time, the mesh sublayer also sets its attributes in the MeshIB, such as *meshPANId*, *meshTreeLevel*, and *meshNetworkAddress*. If no suitable channel or PAN identifier can be found, the MHME issues the MHME-START-NETWORK.confirm primitive with the Status parameter set to STARTUP_FAILURE.

If only a single channel is provided in the higher layer request, the MHME does not need to request an energy scan prior to starting the network. An active scan is still conducted to ensure a PAN identifier is selected that does not conflict with that of any network known to be operating.

Next, the MHME issues the MLME-START.request primitive to the MAC layer. The PANCoordinator parameter of the MLME-START.request primitive is set to TRUE. The BeaconOrder and SuperframeOrder will be the same as those given to the MHME-START-NETWORK.request. The CoordRealignment parameter in the MLME-START.request primitive is set to FALSE if the primitive is issued to start a new PAN. The CoordRealignment parameter is set to TRUE if the primitive is issued to change any of the PAN configuration attributes on an existing PAN. On receipt of the associated MLME-START.confirm primitive, the MHME issues the MHME-START-NETWORK.confirm primitive to the next higher layer with the status returned from the MLME-START.confirm primitive.

5.2.2.4 MHME-START-NETWORK.confirm

This primitive reports to the next higher layer the result of the request to initialize a mesh coordinator in a network.

The semantics of the primitive is as follows:

```
MHME-START-NETWORK.confirm      (
                                    Status
                                )
```

Table 12 specifies the parameters for the MHME-START-NETWORK.confirm primitive.

Table 12—MHME-START-NETWORK.confirm parameters

Name	Type	Valid range	Description
Status	Status	INVALID_REQUEST, STARTUP_FAILURE or any status value returned from the MLME-START.confirm primitive	The result of the attempt to initialize a mesh coordinator

This primitive is generated by the MHME and issued to its next higher layer in response to an MHME-START-NETWORK.request primitive. The primitive returns a status value of INVALID_REQUEST, STARTUP_FAILURE or any status value returned from the MLME-START.confirm primitive. Conditions under which these values may be returned are described in 5.2.2.3.

On receipt of this primitive, the next higher layer is notified of the result of its request to initialize the device as a mesh coordinator. If the MHME has been successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

5.2.2.5 MHME-START-DEVICE.request

This primitive allows the next higher layer of a mesh device to initiate the activities expected of a mesh device, including the routing of data frames and the accepting of requests to join the network from other devices.

The semantics of the primitive is as follows:

```
MHME-START-DEVICE.request  (
                             BeaconOrder,
                             SuperframeOrder
                           )
```

Table 13 specifies the parameters for MHME-START-DEVICE.request.

Table 13—MHME-START-DEVICE.request parameters

Name	Type	Valid range	Description
BeaconOrder	Integer	0x0f	The beacon order of the network that the higher layers wish to form. For this version of the recommended practice, the value is always set to 0x0f indicating no periodic beacons are transmitted.
SuperframeOrder	Integer	0x0f	The superframe order of the network that the higher layers wish to form. For this version of the recommended practice, the value is always set to 0x0f indicating no periodic beacons are transmitted.

This primitive is generated by the next higher layer of a new device and issued to its MHME to request the initialization of itself as a mesh device.

On receipt of this primitive by a device that is not already joined to a mesh network as a mesh device, the MHME issues the MHME-START-DEVICE.confirm primitive with the Status parameter set to INVALID_REQUEST.

On receipt of this primitive by the MHME of a device that is joined to a mesh network as a mesh device, the MHME issues the MLME-START.request primitive to the MAC sublayer. The BeaconOrder and SuperframeOrder parameters of the MLME-START.request primitive will have the values given by the corresponding parameters of the MHME-START-DEVICE.request. The PANCoordinator parameter in the MLME-START.request primitive is set to FALSE indicating this device is not the PAN coordinator of the network. The CoordRealignment parameter in the MLME-START.request primitive is set to FALSE if the primitive is issued to start as a mesh device for the first time. The CoordRealignment parameter is set to TRUE thereafter if the primitive is issued to change any of the PAN configuration attributes.

On receipt of the associated MLME-START.confirm primitive, the MHME issues the MHME-START-DEVICE.confirm primitive to the next higher layer with the status returned from the MLME-START.confirm primitive. If, and only if, the status returned from the MLME-START.confirm primitive is SUCCESS, the device may then begin to engage in the routing of data frames and the accepting of requests to join the network from other devices. Otherwise, the device shall not engage in these activities.

5.2.2.6 MHME-START-DEVICE.confirm

This primitive reports the results of the request to initialize a mesh device.

The semantics of the primitive is as follows:

```
MHME-START-DEVICE.confirm (
                          Status
                          )
```

Table 14 specifies the parameters for MHME-START-DEVICE.confirm.

Table 14—MHME-START-DEVICE.confirm parameter

Name	Type	Valid range	Description
Status	Status	INVALID_REQUEST or any status value returned from the MLME-START.confirm primitive	The result of the attempt to initialize a mesh device

This primitive is generated by the MHME and issued to its next higher layer in response to an MHME-START-MESH-DEVICE.request primitive. This primitive return a status value of INVALID_REQUEST or any status value returned from the MLME-START.confirm primitive. Conditions under which these values may be returned are described in 5.2.2.5.

On receipt of this primitive, the next higher layer is notified of the results of its request to initialize a mesh device. If the MHME has been successful, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

5.2.2.7 MHME-JOIN.request

This primitive allows the next higher layer of a device to request to join or rejoin a network.

The semantics of the primitive is as follows:

```
MHME-JOIN.request    (
                     DirectJoin,
                     ParentDevAddr,
                     PANId,
                     RejoinNetwork,
                     JoinAsMeshDevice,
                     ScanChannels,
                     ScanDuration,
                     ChannelPage,
                     CapabilityInformation
                     )
```

Table 15—MHME-JOIN.request parameters

Name	Type	Valid range	Description
DirectJoin	Boolean	TRUE or FALSE	The value is set to TRUE if direct joining is chosen; otherwise, its value is FALSE.
ParentDevAddr	Integer	0x0000–0xffff	The 16-bit short address of the parent device to join. This field will be read only when the DirectJoin parameter has a value equal to TRUE.
PANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the network to join. This field will be read only when the DirectJoin parameter has a value equal to TRUE.
RejoinNetwork	Integer	0x00–0x01	This parameter controls the method of joining the network. The parameter is 0x00 if the device is requesting to join a network through association. The parameter is 0x01 if the device is joining the network using the mesh sublayer rejoining procedure. This field will be read only when the DirectJoin parameter has a value equal to TRUE.
JoinAsMeshDevice	Boolean	TRUE or FALSE	The parameter is set to TRUE if the device is going to function as a mesh device; it is set to FALSE if the device is going to function as an end device.

Table 15—MHME-JOIN.request parameters *(continued)*

Name	Type	Valid range	Description
ScanChannels	Bit field	27-bit field	The 27 bits (b_0, b_1,... b_{26}) indicate which channels are to be scanned (1 = scan, 0 = do not scan) for each of the 27 channels supported by the ChannelPage parameter. This field will be read only when the DirectJoin parameter has a value equal to FALSE.
ScanDuration	Integer	0–14	A value used to calculate the length of time to spend scanning each channel: The time spent scanning each channel is (*aBaseSuperframeDuration* * (2^n + 1)) symbols, where *n* is the value of the ScanDuration parameter. For more information on MAC sublayer scanning, please refer to IEEE Std 802.15.4-2006. This field will be read only when the DirectJoin parameter has a value equal to FALSE.
ChannelPage	Integer	0–31	The channel page on which to perform the scan. This field will be read only when the DirectJoin parameter has a value equal to FALSE.
CapabilityInformation	Boolean fields	As described in Table 16	The operating capabilities of the device being directly joined.

Table 16—Capability Information bit-fields

Name	Type	Valid range	Description
DeviceType	Boolean	TRUE, FALSE	This field is set to TRUE if the joining device is a mesh device. Otherwise, it is set to FALSE.
PowerSource	Boolean	TRUE, FALSE	This field is set to TRUE if it is mains-powered. Otherwise, it is set to FALSE.
ReceiverOnWhenIdle	Boolean	TRUE, FALSE	This field is set to TRUE if the receiver is enabled when the device is idle. Otherwise, it is set to FALSE.
AllocateAddress	Boolean	TRUE, FALSE	This field is always set to TRUE in the implementations of this recommended practice, indicating that the joining device should be issued a 16-bit network address.

The next higher layer of a device generates this primitive to request to:

— Join a new network
— Locate and rejoin a network after being disconnected

On receipt of this primitive by a device that is currently joined to a network and with the RejoinNetwork parameter equal to 0x00, the MHME issues an MHME-JOIN.confirm primitive with the status parameter set to INVALID_REQUEST.

On receipt of this primitive by a device that is not currently joined to a network and with the RejoinNetwork parameter equal to 0x00, the device attempts to join the network. If the DirectJoin parameter is set to TRUE, the device attempts to join the parent device whose address is specified by the ParentDevAddr field. The MHME issues an MLME-ASSOCIATE.request with its CoordAddress parameter set to ParentDevAddr if the following occur:

— the parent device belongs to the network identified by the PANId, and

— the parent device is open to join requests and is advertising capacity of the correct device type.

If the DirectJoin parameter is set to FALSE, the device attempts to join the network through a neighbor device found by the discovery procedure. The MHME issues an MLME-ASSOCIATE.request with its CoordAddress parameter set to the address of a mesh device for which the following conditions are true:

— The mesh device belongs to the network identified by the PANId

— The mesh device is open to join requests and is advertising capacity of the correct device type

— The link quality for frames received from this device is such that a link quality of at least 128 is produced (the link quality ranges from 0, the lowest, to 255, the highest)

If a device exists in the mesh discovery table as illustrated in Table 9 for which these conditions are true, the LogicalChannel parameter of the MLME-ASSOCIATE.request primitive is set to that found in the MDT, corresponding to the coordinator address of the potential parent. The fields of the CapabilityInformation parameter should have the values shown in Table 16 and the capability information assembled here should be stored as the value of the meshCapabilityInformation MeshIB attribute. If more than one device meets these requirements, then the joining device may select the parent with the smallest tree level or/and highest link quality. How to weight the tree level and link quality is not defined in this document. The implementers are given the flexibility to choose the way that meets their application requirements best.

If no device exists in the mesh discovery table for which the conditions are true, the mesh sublayer should issue an MHME-JOIN.confirm with the Status parameter set to NOT_PERMITTED. Otherwise, the MHME issues the MHME-JOIN.confirm with the Status parameter set to the status parameter value returned from the MLME-ASSOCIATE.confirm primitive.

If the RejoinNetwork parameter is 0x00 and the JoinAsMeshDevice parameter is set to TRUE, the device will function as a mesh device in the network. If the JoinAsMeshDevice parameter is FALSE, then it will join as an end device and not participate in routing.

If a device receives this primitive and the RejoinNetwork parameter is equal to 0x01 then it should join the network using the mesh sublayer rejoin process as described in 5.5.11.2.

5.2.2.8 MHME-JOIN.indication

This primitive allows the next higher layer of a mesh coordinator or a mesh device to be notified when a new device has successfully joined its network by association or rejoined using mesh sublayer rejoin procedure.

The semantics of the primitive is as follows:

```
MHME-JOIN.indication (
                     NetworkAddress,
                     ExtendedAddress,
                     CapabilityInformation,
                     RejoinNetwork
                     )
```

Table 17 specifies the parameters for the MHME-JOIN.indication primitive.

Table 17—MHME-JOIN.indication parameters

Name	Type	Valid range	Description
NetworkAddress	Network address	0x0001–0xfffd, or 0xfffe	When a short network address of an entity has been assigned, this will be the short address that has been added to the network. Otherwise, the value of this parameter will equal to 0xfffe indicating the short address has not been assigned and the device can only be reached by its 64-bit extended address.
ExtendedAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address of an entity that has been added to the network.
CapabilityInformation	Bit field	As defined in IEEE Std 802.15.4-2006.	Specifies the operational capabilities of the joining device.
RejoinNetwork	Integer	0x00–0x02	The RejoinNetwork parameter indicating the method used to join the network. The parameter is 0x00 of the device joined through association. The parameter is 0x01 if the device joined through mesh sublayer rejoin precedure.

This primitive is generated by the MHME of a mesh coordinator or mesh device and is issued to its next higher layer on successfully adding a new device to the network using the MAC association, or on allowing a device to rejoin the network using the rejoining procedure. Note when the NetworkAddress parameter has a value equal to 0xfffe, it means the network is in the initialization state and the short address has not been assigned to the joining device yet. This device should later assign the addresses to all its child devices through the address assignment command frame as defined in Figure 11.

On receipt of this primitive, the next higher layer of a mesh coordinator or mesh device is notified that a new device has joined its network.

5.2.2.9 MHME-JOIN.confirm

This primitive allows the next higher layer to be notified of the result of its request to join a network.

The semantics of the primitive is as follows:

```
MHME-JOIN.confirm    (
                     Status,
                     NetworkAddress,
                     PANId,
                     ChannelPage,
                     ActiveChannel
                     )
```

Table 18 specifies the parameters for the MHME-JOIN.confirm primitive.

Table 18—MHME-JOIN.confirm parameters

Name	Type	Valid range	Description
Status	Status	INVALID_REQUEST, NOT_PERMITTED, NO_NETWORKS or any status value returned from the MLME-ASSOCIATE.confirm primitive or the MLME-SCAN.confirm primitive.	The status of the corresponding request.
NetworkAddress	Integer	0x0001–0xffff	The 16-bit network address that was allocated to this device. This parameter will be equal to 0xffff if the join attempt was unsuccessful.
PANId	Integer	0x0000–0xfffe	The 16-bit PAN identifier of the network of which the device is now a member.
ChannelPage	Integer	0–31	The channel page on which the ActiveChannel was found.
ActiveChannel	Integer	The number of any channel supported by the PHY as defined in IEEE Std 802.15.4-2006.	The value of phyCurrentChannel parameter which is equal to the current channel of the network that has been joined.

This primitive is generated by the initiating MHME and issued to its next higher layer in response to an MHME-JOIN.request primitive.

If the request was not successful, this primitive should be sent to the next higher layer as soon as the status of the join request is known. The status parameter should indicate an error code of INVALID_REQUEST, NOT_PERMITTED, NO_NETWORKS or any status value returned from either the MLME-ASSOCIATE.confirm primitive or the MLME-SCAN.confirm primitive. The reasons for these status values are fully described in 5.2.2.8.

If the request was successful, this primitive is not sent to the next higher layer until a short address is successfully assigned by receiving an address assignment command frame from the parent device. Whether or not the joining device can obtain a short address immediately by using this join request depends on the operating states (initialization or operation) of its parent device. In the case its parent device is in the operation state, this device can get a short address from the parent device immediately. Otherwise, this device is just associated with the parent device using its 64-bit extended address. Devices in this status should use their 64-bit extended addresses to report their child numbers to parent devices. The short address will be assigned after the parent device gets the address block assignment from its own parent. Therefore, the mesh sublayer should hold the confirmation message until the short address is assigned from the parent device. Note that the next higher layer is not capable of transmitting any data until the confirmation message is received.

On receipt of this primitive, the next higher layer of the initiating device is notified of the result of its request to join a network using the MAC sublayer association procedure, or to rejoin a network once it has temporarily left.

5.2.2.10 MHME-LEAVE.request

This primitive allows the next higher layer to request that it or a child device leaves the network.

The semantics of the primitive is as follows:

```
MHME-LEAVE.request   (
                     DeviceAddress,
                     RemoveSelf,
                     RemoveChildren
                     )
```

Table 19 specifies the parameters for the MHME-LEAVE.request primitive.

Table 19—MHME-LEAVE.request parameters

Name	Type	Valid range	Description
DeviceAddress	Device address	Any 64-bit, IEEE address	The 64-bit IEEE address of a child device to be removed from the network.
RemoveSelf	Boolean	TRUE, FALSE	This parameter has a value of TRUE if the device is asked to remove itself from the network. Otherwise it has a value of FALSE.
RemoveChildren	Boolean	TRUE, FALSE	This parameter has a value of TRUE if the device being asked to leave the network is also being asked to remove its child devices, if any. Otherwise it has a value of FALSE.

The next higher layer of a device generates this primitive to request to leave the network. The next higher layer of a mesh coordinator or mesh device may also generate this primitive to remove a device from the network.

On receipt of this primitive by the MHME of a device that is not currently joined to a network, the MHME issues the MHME-LEAVE.confirm primitive with a status of INVALID_REQUEST. On receipt of this primitive by the MHME of a device that is currently joined to a network, with the RemoveSelf parameter set to TRUE and the RemoveChildren parameter set to FALSE, the MHME will remove itself from the network as described in 5.5.7.1. Basically, the MHME should send out a hello command frame to its neighbors to indicate its leaving and clear all of its references to the mesh network. It should then issue an MLME-RESET.request primitive to the MAC sublayer. If the MHME-LEAVE.request primitive is received with the RemoveSelf parameter set to TRUE and the RemoveChildren parameter equal to TRUE, then the MHME should first attempt to remove the device's children before removing itself from the network.

On receipt of this primitive by a mesh coordinator or mesh device and with the RemoveSelf parameter set to FALSE, the MHME determines whether the specified device is a child device. If the requested device does not exist, the MHME issues the MHME-LEAVE.confirm primitive with a status of UNKNOWN_CHILD_DEVICE. If the requested device exists, the MHME will attempt to remove it from the network following the procedure described in 5.5.7.2. Basically the MHME should first send a leave command to the requested device and the device will remove itself following the procedure described in 5.5.7.1. If the RemoveChildren parameter is equal to TRUE then the device will be requested to remove its children as well. Upon receiving the acknowledgement of the leave command, the MHME will issue the MHME-LEAVE.confirm primitive with the DeviceAddress equal to the 64-bit IEEE address of the removed device and the Status parameter equal to the status delivered by the MCPS-DATA.confirm primitive.

5.2.2.11 MHME-LEAVE.indication

This primitive allows the next higher layer of a device to be notified if that device has left the network or if a neighboring device has left the network.

The semantics of the primitive is as follows:

```
MHME-LEAVE.indication    (
                         DeviceAddress
                         )
```

Table 20 specifies the parameters for the MHME-LEAVE.indication primitive.

Table 20—MHME-LEAVE.indication parameters

Name	Type	Valid range	Description
DeviceAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address of an entity that has removed itself from the network

This primitive is generated by the MHME of a mesh device and issued to its next higher layer on the receipt of a leave command asking it to leave the network. It is also generated by the MHME of a mesh device and issued to its next higher layer to indicate that a neighbor has left the network upon receiving a hello command frame with the leaving network flag set to one.

On receipt of this primitive, the next higher layer of a mesh coordinator or mesh device is notified that a device, formerly on the network, has left. The primitive can also indicate that the next higher layer of a mesh device or end device is informed that it has been removed from the network by its associated mesh device or mesh coordinator.

5.2.2.12 MHME-LEAVE.confirm

This primitive allows the next higher layer to be notified of the results of its request for itself or another device to leave the network.

The semantics of the primitive is as follows:

```
MHME-LEAVE.confirm  (
                    Status,
                    DeviceAddress
                    )
```

Table 21 specifies the parameters for the MHME-LEAVE.confirm primitive.

Table 21—MHME-LEAVE.confirm parameters

Name	Type	Valid range	Description
Status	Status	SUCCESS, INVALID_REQUEST, UNKNOWN_CHILD_DEVICE or any status returned by the MCPS-DATA.confirm primitive	The status of the corresponding request
DeviceAddress	64-bit IEEE address	Any 64-bit, IEEE address	The 64-bit IEEE address in the request to which this is a confirmation or null if the device requested to remove itself from the network

This primitive is generated by the initiating MHME and issued to its next higher layer in response to an MHME-LEAVE.request primitive. If the request was successful, the status parameter indicates a successful leave attempt. Otherwise, the status parameter indicates an error code of INVALID_REQUEST, UNKNOWN_CHILD_DEVICE or a status delivered by the MCPS-DATA.confirm primitive. The reasons for these status values are fully described in 5.2.2.10.

On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its request for itself or a child device to leave the network.

5.2.2.13 MHME-RESET.request

This primitive allows the next higher layer to request that the mesh sublayer performs a reset operation.

The semantics of the primitive is as follows:

```
MHME-RESET.request    (
                      )
```

This primitive has no parameters.

This primitive is generated by the next higher layer and issued to its MHME to request the reset of the mesh sublayer to its initial condition.

On receipt of this primitive, the MHME first checks whether it is connected to any networks. If it is connected a network, the MHME should broadcasts a hello command frame to indicate its leaving to its neighbors up to *meshTTLOfHello* hops. If it is not connected to any networks, the MHME should skip the checking and go directly to issue the MLME-RESET.request primitive to the MAC sublayer with the SetDefaultPIB parameter set to TRUE. On receipt of the corresponding MLME-RESET.confirm primitive, the mesh sublayer resets it by clearing all internal variables, mesh discovery table entries and by setting all MeshIB attributes to their default values. Once the mesh sublayer is reset, the MHME issues the MHME-RESET.confirm with the Status parameter set to SUCCESS if the MAC sublayer was successfully reset.

If this primitive is issued to the MHME of a device that is currently joined to a network, any required leave attempts using the MHME-LEAVE.request primitive should be made a priority at the discretion of the next higher layer.

5.2.2.14 MHME-RESET.confirm

This primitive allows the next higher layer to be notified of the results of its request to reset the mesh sublayer.

The semantics of the primitive is as follows:

```
MHME-RESET.confirm    (
                                    Status
                      )
```

Table 22 specifies the parameters for this primitive.

Table 22—MHME-RESET.confirm parameter

Name	Type	Valid range	Description
Status	Status	Any status value returned from the MLME-RESET.confirm primitive as defined in IEEE Std 802.15.4-2006.	The result of the reset operation

This primitive is generated by the MHME and issued to its next higher layer in response to an MHME-RESET.request primitive. If the request was successful, the Status parameter will have a value of SUCCESS. Otherwise, the status parameter will have an error code indicating the reason of failure. The reasons for these status values are fully described in 5.2.2.13.

5.2.2.15 MHME-GET.request

This primitive allows the next higher layer to read the value of an attribute from the MeshIB, which is defined in Table 42.

The semantics of the primitive is as follows:

```
MHME-GET.request    (
                    MIBAttribute
                    )
```

Table 23 specifies the parameters for this primitive.

Table 23—MHME-GET.request parameter

Name	Type	Valid range	Description
MeshIB Attribute	Integer	As described in Table 42	The identifier of the MeshIB attribute to read

On receipt of the MHME-GET.request primitive, the MHME checks to see if the MeshIB Attribute is a MeshIB attribute or a MAC/PHY PIB attribute. If the requested attribute is a MeshIB attribute, the MHME attempts to retrieve the requested MeshIB attribute from its database. If the identifier of the MeshIB attribute is not found in the database, the MHME will issue the MHME-GET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the requested MeshIB attribute is successfully retrieved, the MHME will issue the MHME-GET.confirm primitive with a status of SUCCESS.

If the requested attribute is a MAC/PHY PIB attribute, the request is passed to the MAC by issuing the MLME-GET.request primitive. The MHME will issue the MHME-GET.confirm primitive to the next higher layer with the status parameter resulting from the translation and the MIBAttribute and MIBAttributeValue parameters equal to those returned by the MLME primitive.

5.2.2.16 MHME-GET.confirm

This primitive reports the results of an attempt to read the value of an attribute from the MIB.

The semantics of the primitive is as follows:

```
MHME-GET.confirm    (
                    Status,
                    MIBAttribute,
                    MIBAttributeLength,
                    MIBAttributeValue
                    )
```

Table 24 specifies the parameters for this primitive.

Table 24—MHME-GET.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, UNSUPPORTED_ATTRIBUTE	The results of the request to read an MeshIB attribute value.
MIBAttribute	Integer	As described in Table 42	The identifier of the MeshIB attribute that was read.
MIBAttributeLength	Integer	0x0000–0xffff	The length, in octets, of the attribute value being returned.
MIBAttributeValue	Various	Attribute specific, as defined in Table 42	The value of the MeshIB attribute that was read.

This primitive is generated by the MHME and issued to its next higher layer in response to an MHME-GET.request primitive. This primitive returns either a status of SUCCESS, indicating that the request to read a MeshIB attribute was successful, or an error code of UNSUPPORTED_ATTRIBUTE. The reasons for these status values are fully described in 5.2.2.15.

On receipt of this primitive, the next higher layer is notified of the results of its request to read a MeshIB attribute.

5.2.2.17 MHME-SET.request

This primitive allows the next higher layer to write the value of an attribute into the MIB.

The semantics of the primitive is as follows:

```
MHME-SET.request    (
                    MIBAttribute,
                    MIBAttributeLength,
                    MIBAttributeValue
                    )
```

Table 25 specifies the parameters for this primitive.

Table 25—MHME-SET.request parameters

Name	Type	Valid range	Description
MIBAttribute	Integer	As described in Table 42	The identifier of the MeshIB attribute to be written.
MIBAttributeLength	Integer	0x0000—0xffff	The length, in octets, of the attribute value being set.
MIBAttributeValue	Various	Attribute specific, as defined in Table 42	The value of the MeshIB attribute that should be written.

This primitive is to be generated by the next higher layer and issued to its MHME to write the value of an attribute in the MIB.

On receipt of the MHME-SET.request primitive, the MHME checks to see if the MIBAttribute is a MeshIB attribute or MAC/PHY PIB attribute. If the requested attribute is a MeshIB attribute, the MHME attempts to write the given value to the indicated MeshIB attribute in its database. If the MIBAttribute parameter specifies an attribute that is a read-only attribute, the MHME will issue the MHME-SET.confirm primitive with a status of READ_ONLY. If the MIBAttribute parameter specifies an attribute that is not found in the database, the MHME will issue the MHME-SET.confirm primitive with a status of UNSUPPORTED_ATTRIBUTE. If the MIBAttributeValue parameter specifies a value that is out of the valid range for the given attribute, the MHME will issue the MHME-SET.confirm primitive with a status of INVALID_PARAMETER. If the requested MeshIB attribute is successfully written, the MHME will issue the MHME-SET.confirm primitive with a status of SUCCESS.

If the requested attribute is a MAC/PHY PIB attribute, the request is passed to the MAC by issuing the MLME-SET.request primitive. The MHME will issue the MHME-SET.confirm primitive to the next higher layer with the status parameter resulting from the translation and the MIBAttribute parameter equal to that returned by the MLME primitive.

5.2.2.18 MHME-SET.confirm

The primitive reports the results of an attempt to write a value to a MeshIB attribute.

The semantics of the primitive is as follows:

```
MHME-SET.confirm    (
                    Status,
                    MIBAttribute
                    )
```

Table 26—MHME-SET.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	SUCCESS, INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE	The results of the request to write the MeshIB attribute.
MIBAttribute	Integer	As described in Table 42	The identifier of the MeshIB attribute that was written.

The primitive is generated by the MHME and issued to its next higher layer in response to an MHME-SET.request primitive. This primitive returns a status of either SUCCESS, indicating that the requested value was written to the indicated MeshIB attribute, or an error code of INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE. The reasons for these status values are fully described in 5.2.2.17.

On receipt of this primitive, the next higher layer is notified of the results of its request to write the value of a MeshIB attribute. If the requested value was written to the indicated MeshIB attribute, the status parameter will be set to SUCCESS. Otherwise, the status parameter indicates the error.

5.2.2.19 MHME-START-SYNC.request

The primitive allows the next higher layer of the mesh coordinator to request the mesh sublayer to start synchronization for synchronous energy saving (SES).

The semantics of the primitive is as follows:

```
MHME-START-SYNC.request     (
                            AO,
                            WO,
                            SyncRegion,
                            SyncInterval
                            )
```

Table 27 specifies the parameters of the MHME-START-SYNC.request primitive.

Table 27—MHME-START-SYNC.request parameters

Name	Type	Valid range	Description
AO	Integer	0–WO or 15	The length of the active portion of the time structure. If the WO has a value of 15, this parameter is ignored. Refer to 5.5.10.2.1 for an explanation of the relationship between the AO and the active duration.
WO	Integer	0–15	The time interval that a device periodically wakeup in energy saving mode. A value of 15 indicates SES will not be used. Refer to 5.5.10.2.1 for an explanation of the relationship between the WO and the wakeup interval.
SyncRegion	Integer	0x00–0xff	The hop distance to be synchronized by a mesh coordinator or region synchronizer during synchronization duration.
SyncInterval	Integer	0x00–0xff	The number of wakeup interval times for periodic synchronization.

This primitive is generated by the next higher layer of the mesh coordinator and issued to its MHME to request the synchronization procedure for SES.

On receipt of this primitive by a device that is not capable of being a mesh coordinator (MC) (by checking the mesh sublayer constant, *meshcCoordinatorCapability*), the MHME issues the MHME-START-SYNC.confirm primitive with the status parameter set to INVALID_REQUEST. Otherwise, the MHME of the mesh coordinator should attempt to start the synchronization procedure by sending a sync request command frame with the parameters AO, WO, SyncRegion, SyncInterval and timestamp in the frame.

The mesh coordinator should transmit two successive synchronization request command frames with timestamp parameters in the frames.

If the MHME of mesh coordinator receives a synchronization reply frame from each of its child devices, it should issue a MHME-START-SYNC.confirm primitive to the next higher layer and start to operate in SES mode from the following wakeup interval.

5.2.2.20 MHME-START-SYNC.confirm

This primitive reports to the next higher layer the results of the synchronization procedure.

The semantics of the primitive is as follows:

```
MHME-START-SYNC.confirm    (
                           Status
                           )
```

Table 28 specifies the parameters of the MHME-START-SYNC.confirm primitive.

Table 28—MHME-START-SYNC.confirm parameters

Name	Type	Valid range	Description
Status	Enumeration	INVALID_REQUEST, SYNC_SUCCESS, SYNC_FAILURE	The results of the attempt to start synchronization procedure using synchronous energy saving scheme.

This primitive is generated by the MHME and issued to its next higher layer in response to an MHME-START-SYNC.request primitive. It is sent when the MHME of mesh coordinator receives a synchronization reply frame from each of its child devices. The primitive returns a status value of INVALID_REQUEST, SYNC_SUCCESS or SYNC_FAILURE depending on the status of the initialization of the synchronization procedure.

5.2.2.21 MHME-TRACE-ROUTE.request

This primitive allows the next higher layer of a mesh device to request the mesh sublayer to trace the route to a specific destination.

The semantics of the primitive is as follows:

```
MHME-TRACE-ROUTE.request    (
                            DestAddr,
                            ResponseTimeout,
                            BatchSize,
                            maxTTL
                            )
```

Table 29 specifies the parameters of the MHME-TRACE-ROUTE.request primitive.

Table 29—MHME-TRACE-ROUTE.request parameters

Name	Type	Valid range	Description
DestAddr	Integer	0x0000–0xfffd	Specifies the mesh address of the destination.
ResponseTimeout	Integer	0–16,777,215	Timeout limitation for trace route reply frames in millisecond.
BatchSize	integer	0–255	Number of trace route request frames to be transmitted for each hop.
maxTTL	Integer	0–255	Maximum number of hops the traceroute request command frame allowed to propagate.

This primitive is generated by the next higher layer of a device and issued to its MHME to trace the route to the destination specified in this primitive.

On receipt of this primitive, the mesh sublayer should issue a traceroute request command frame toward the destination, using the provided parameters.

5.2.2.22 MHME-TRACE-ROUTE.indication

This primitive allows the next higher layer of a mesh device to be notified when a traceroute reply command frame has been successfully received. It returns round trip times between the originator of traceroute request to an intermediate mesh device or the destination device.

The semantics of the primitive is as follows:

```
MHME-TRACE-ROUTE.indication (
                            Status,
                            SrcAddr,
                            RTTStamp,
                            HopCount
                            )
```

Table 30 specifies the parameters of the MHME-TRACE-ROUTE.indication primitive.

Table 30—MHME-TRACE-ROUTE.indication parameters

Name	Type	Valid range	Description
Status	Boolean	TRUE, FALSE	If TRUE, indicates successful reception of traceroute reply frame within ResponseTimeout.
SrcAddr	Integer	0x0000–0xfff7	Specifies the mesh address of the source of the traceroute reply frame.
RTTStamp	Integer	0–65535	Time duration, in milliseconds, between the traceroute requests is sent to the traceroute reply frame is received.
HopCount	Integer	0x00–0x0f	The hop count between the originator of the traceroute reply and this device.

This primitive is generated by the mesh sublayer of a device and issued to its next higher layer.

On receipt of this primitive, the next higher layer is notified the result of current step of trace route.

5.2.2.23 MHME-TRACE-ROUTE.confirm

This primitive is used to report the final result of the MHME-TRACE-ROUTE request.

The semantics of the primitive is as follows:

```
MHME-TRACE-ROUTE.confirm  (

                          DestAddr,
                          Status
                          )
```

Table 31 specifies the parameters of the MHME-TRACE-ROUTE.confirm primitive.

Table 31 — MHME-TRACE-ROUTE.confirm parameters

Name	Type	Valid range	Description
DestAddr	Integer	0x0000–0xfff7	Specifies the mesh address of the destination of the traceroute request command frame.
Status	Boolean	TRUE, FALSE	If TRUE, indicates successful reception of traceroute reply from the destination within *maxTTL* specified by the MHME-TRACE-ROUTE.request primitive.

This primitive is generated by the mesh sublayer of a device and issued to its next higher layer.

On receipt of this primitive, the next higher layer is notified the final result of the trace route. It also indicates the trace route process is completed.

5.2.2.24 MHME-MULTICAST-JOIN.request

This primitive allows the next higher layer of a device to request the MHME to join a multicast group.

The semantics of the primitive is as follows:

```
MHME-MULTICAST-JOIN.request     (
                                GroupAddress,
                                GCAddress,
                                JoinAsGC
                                )
```

Table 32 — MHME-MULTICAST-JOIN.request parameters

Name	Type	Valid range	Description
GroupAddress	Integer	0x0001–0xfffe	Indicates the address of the multicast group the device intends to join.
GCAddress	Integer	0x0001–0xffff	The unicast address of the group coordinator of this multicast group. Value 0xffff indicates the GC's unicast address is not known or not provided.
JoinAsGC	Boolean	TRUE, FALSE	Indicates if the device is joining as the group coordinator.

This primitive is generated by the next higher layer of a device and issued to its MHME to request joining a multicast group specified by the GroupAddress parameter.

Upon receiving this primitive, the MHME of the device checks whether the device has already joined the multicast group. If it is already a member of the group, the MHME issues a MHME-MULTICAST-JOIN.confirm primitive with the Status set to INVALID_REQUEST.

If it is not yet a member of the group, it should check whether it is an end device. If it is an end device, the MHME should issue a group join request (G-JREQ) command toward its parent device with the end device bit of the Join Options field set to indicate an end device. The parent device should then join the multicast group on behalf of its end device.

If it is not yet a member of the group and not an end device, it checks to see whether the JoinAsGC parameter is set. If the parameter is set to TRUE, the MHME should issue a G-JREQ command toward the mesh coordinator as described in 5.5.8.2.1 If the parameter is set to FALSE, the MHME should issue a G-JREQ command following the process described in 5.5.8.2.1. The GCAddress parameter, if provided, can be used to route the join request command frame toward the direction of the GC.

5.2.2.25 MHME-MULTICAST-JOIN.confirm

This primitive allows the MHME of the device to report the result of joining a multicast group.

The semantics of the primitive is as follows:

```
MHME-MULTICAST-JOIN.confirm (
                            Status
                            )
```

This primitive is generated by the MHME and issued to the next higher layer of a device in response to an MHME-MULTICAST-JOIN.request primitive.

If the request was not successful, this primitive should be sent to the next higher layer as soon as the status of the join request is known. The status parameter should indicate an error code of INVALID_REQUEST or GC_NOT_YET_JOINED.

If a group join reply (G-JREP) command frame is received in response to a G-JREQ command frame, the MHME should issue to its next higher layer a MHME-MULTICAST-JOIN.confirm primitive with the status set to SUCCESS.

On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its request to join a multicast group.

5.2.2.26 MHME-MULTICAST-LEAVE.request

This primitive allows the next higher layer of a device to request the MHME to leave a multicast group.

The semantics of the primitive is as follows:

```
MHME-MULTICAST-LEAVE.request (
                             GroupAddress
                             )
```

Table 33— MHME-MULTICAST-LEAVE.request parameters

Name	Type	Valid range	Description
GroupAddress	Integer	0x0001–0xfffe	The address of the multicast group to leave

This primitive is generated by the next higher layer of a device and issued to its MHME to request to leave a multicast group specified by the group address parameter.

Upon receiving this primitive, the MHME of the device should check whether the device is a member of the multicast group. If it is not a member of the group, the MHME issues a MHME-MULTICAST-LEAVE.confirm primitive with the status set to INVALID_REQUEST.

If it is a member of the group, the MHME of the device should then check whether the device is an end device. If it is an end device, it should leave the multicast group by sending a group leave request (G-LREQ) command frame to its parent device.

If it is neither a member of the group nor an end device, it should attempt to leave the group by issuing a hello command frame with the multicast related fields indicating a leave. The multicast leaving process is described in details in 5.5.8.2.2.

5.2.2.27 MHME-MULTICAST-LEAVE.confirm

This primitive is used to report the result of leaving a multicast group.

The semantics of the primitive is as follows:

```
MHME-MULTICAST-LEAVE.confirm        (
                                      Status
                                    )
```

This primitive is generated by the MHME and issued to the next higher layer of the mesh coordinator or a mesh device in response to an MHME-MULTICAST-LEAVE.request primitive.

If the request was not successful, this primitive should be sent to the next higher layer as soon as the status of the join request is known. The status parameter should indicate an error code of INVALID_REQUEST or any valid status values.

If the leave process was successful, the MHME should issue to its next higher layer a MHME-MULTICAST-LEAVE.confirm primitive with the status set to SUCCESS.

On receipt of this primitive, the next higher layer of the initiating device is notified of the results of its request to leave a multicast group.

5.3 Frame formats

5.3.1 General mesh service frame format

The general mesh service frame should be formatted as illustrated in Figure 4. Some subfields of a field might be reserved for future use, and they should be set to zero for all implementations based on this version of recommended practice. For every mesh service frame, all reserved bits should be set to zero

upon transmission and should be ignored upon receipt. Mesh service frames will be passed to the MAC layer and to be carried in the payload of MAC data frames.

<table>
<tr><th>Octets: 2</th><th>8/2</th><th>8/2</th><th>Variable</th></tr>
<tr><td>Frame Control</td><td>Destination Address</td><td>Source Address</td><td rowspan="2">Mesh Sublayer payload</td></tr>
<tr><td colspan="3">Mesh Sublayer Header</td></tr>
</table>

Figure 4— General mesh service frame format

5.3.1.1 Frame Control field

The Frame Control field should be formatted as illustrated in Figure 5.

Bits: 0–3	4	5	6	7–10	11–15
Protocol Version	Frame Type	Destination Address Mode	Source Address Mode	Transmission Options	Reserved

Figure 5— Frame Control field

5.3.1.1.1 Protocol Version subfield

The Protocol Version subfield should be set to one for all implementations based on this version of recommended practice.

5.3.1.1.2 Frame Type subfield

The valid values for the Frame type field are given in Table 34.

Table 34— Valid values of Frame Type subfield

Frame type value	Frame type name
0	Data frame
1	Command frame

5.3.1.1.3 Destination and Source Address Mode subfields

Destination and Source Address Mode subfields indicate whether the 64-bit extended addresses or the 16-bit short addresses are used in the Destination and Source Address fields. The subfields should be set to zero when 64-bit extended addresses are used and should be set to one when 16-bit short addresses are used.

5.3.1.1.4 Transmission Options subfield

The Transmission Options subfield indicates the way the frame should be transmitted. The transmit option parameters, AckTransmission, McstTransmission, BcstTransmission and ReliableBcst, of the MESH-DATA.request primitive should be used to fill in this field in the data frame. This information will be transmitted with the data payload toward the destination so that all intermediate devices know how to transmit the data frame at both the mesh sublayer and the MAC layer.

Bit	Transmission options
b_7	Acknowledged transmission
b_8	Multicast transmission
b_9	Broadcast transmission
b_{10}	Reliable broadcast

Figure 6— Valid values of Transmission Options subfield

5.3.1.2 Destination Address field

Destination Address is either the 64-bit extended address or the 16-bit short address as indicated by the Destination Address Mode subfield of the Frame Control field.

5.3.1.3 Source Address field

Source Address is either the 64-bit extended address or the 16-bit short address as indicated by the Source Address Mode subfield of the Frame Control field.

5.3.1.4 Mesh sublayer payload

Mesh sublayer payload may contain different subfields and be in various lengths depending on the frame type field defined in Table 34 The details are described in 5.3.2.

5.3.2 Format of individual frame types

5.3.2.1 Data frame

The data frame mesh sublayer payload should be formatted as illustrated in Figure 7.

Octets: 2	**8/2**	**8/2**	**1**	**1**	**Variable**
Frame Control	Destination Address	Source Address	Sequence Number	Routing Control	Data Payload
Mesh Sublayer Header			Mesh Sublayer Payload		

Figure 7—Format of the data frame

The details of the Routing Control field are shown in Figure 8.

b_7	**$b_{6–0}$**
Up-down Flag	Reserved

Figure 8—Format of the Routing Control field

The Up-down Flag is used to detect routing loops and link state mismatches as described in 5.5.5.1. The calculation of its value is illustrated in 5.5.6.1.

5.3.2.2 Command frame

The general command frame should be formatted as illustrated in Figure 9.

Octets: 2	**Variable**	**1**	**Variable**
Frame Control	Addressing fields	Command Frame Identifier	Command payload
Mesh Sublayer Header		Mesh Sublayer Payload	

Figure 9—Format of the command frame

For all command frames, the Protocol Version subfield of the Frame Control field should have a value of one. The Frame Type subfield should also have a value of one indicating a command frame.

The details of the Command Frame Identifier field are illustrated in Table 35.

Table 35— Valid values of command frame identifiers field

Command frame identifier	Command name	Mandatory / Optional	Subclause
0x01	Children number report	Mandatory	5.3.2.2.1
0x02	Address assignment	Mandatory	5.3.2.2.2
0x03	Hello	Mandatory	5.3.2.2.3
0x04	Neighbor information request	Mandatory	5.3.2.2.4
0x05	Neighbor information reply	Mandatory	5.3.2.2.5
0x06	Link state	Mandatory	5.3.2.2.6
0x07	Link state mismatch	Mandatory	5.3.2.2.7
0x08	Probe	Mandatory	5.3.2.2.8
0x09	G-JREQ	Mandatory	5.3.2.2.9
0x0a	G-JREP	Mandatory	5.3.2.2.10
0x0b	G-LREQ	Mandatory	5.3.2.2.11
0x0c	Group leave reply	Mandatory	5.3.2.2.12
0x0d	Wakeup notification	Optional	5.3.2.2.13
0x0e	Extension request	Optional	Figure 24
0x0f	Extension reply	Optional	5.3.2.2.15
0x10	Synchronization request	Optional	5.3.2.2.16
0x11	Synchronization reply	Optional	5.3.2.2.17
0x12	Reservation request	Optional	5.3.2.2.18
0x13	Reservation reply	Optional	5.3.2.2.19
0x14	Rejoin notify	Optional	5.3.2.2.20
0x15	Traceroute request	Optional	5.3.2.2.21
0x16	Traceroute reply	Optional	5.3.2.2.22
0x17	Leave	Mandatory	5.3.2.2.23
0x18–0xff	Reserved	–	–

5.3.2.2.1 Children number report command

The children number report command is used for a child device to report its number of child devices to its parent device. The payload of the command frame should be formatted as illustrated in Figure 10.

1	2	2
Command Frame Identifier	Number of Descendants	Number of Requested Addresses
Data Frame Mesh Sublayer Payload		

Figure 10 — Children number report command frame payload format

In the Frame Control field, both Destination Address Mode and Source Address Mode subfields should have values of zero indicating the 64-bit extended addresses, and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address field should contain the 64-bit extended address of the destination, which should be the parent of the source. The Source Address field should contain the MAC address of the source.

The Number of Descendants field should contain the total number of devices along the tree branch starting from the source (including the source itself).

The Number of Requested Addresses field should contain the total number of addresses the source requests. This value should be equal to or larger than the value of the number of descendants.

5.3.2.2.2 Address assignment command

The address assignment command is used for a parent device to assign an address block to its child devices. The payload of the address assignment frame should be formatted as illustrated in Figure 11.

1	2	2	2
Command Frame Identifier	Beginning Address	Ending Address	Tree Level of Parent Device
Data Frame Mesh Sublayer Payload			

Figure 11 — Address assignment command frame payload format

In the Frame Control field, the Destination Address mode subfield should have a value of zero indicating the 64-bit extended address and the Source Address Mode field should have a value of one indicating the 16-bit short address. The Transmission Options subfield should have corresponding values indicating this frame should be unicast and acknowledgement is requested.

The Destination Address field should contain the 64-bit extended address of the destination, which should be a child of the source. The Source Address field should contain the 16-bit short address of the source.

The Beginning Address field should contain the beginning address of the address block assigned to the child.

The Ending Address field should contain the ending address of the address block assigned to the child. Assigned addresses are also referred to as short addresses or logical addresses in this recommended practice.

The Tree Level of Parent Device field should contain the tree level of the parent device which sends this command frame. The receiver of this command frame is able to derive its own tree level (one level more than its parent's tree level) from the value of this field.

5.3.2.2.3 Hello command

The hello command is used to exchange connectivity information among neighbors and to manage network and multicast group membership. The payload of the command frame should be formatted as illustrated in Figure 12.

Octets: 1	**1**	**2**	**2**	**2**	**1**	**1**	**1**	**Variable**	**Variable**
Command Frame Identifier	TTL	Beginning Address	Ending Address	Tree Level	Hello Control	Number of One-hop Neighbors	Number of Multicast Groups	Addresses of One-hop Neighbors	Addresses of Multicast Groups
Data Frame Mesh Sublayer Payload									

Figure 12—Hello command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be broadcast and acknowledgement is not requested.

The Destination Address field should contain the mesh sublayer broadcast address. The Source Address field should contain the mesh address of the source.

The TTL field should contain the number of hops the hello command frame is allowed to propagate. This information can be obtained from the *meshTTLOfHello* attribute of the MeshIB, as defined in Table 42.

The Beginning Address field should contain the beginning address of the source's address block.

The Ending Address field should contain the ending address of the source's address block.

The Tree Level field should contain the tree level as defined by the *meshTreeLevel* attribute of MeshIB in Table 42 of the source.

The Hello Control field is illustrated in Table 36.

Table 36—Valid values of the Hello Control subfield

Bit	Control	Description
b_7	Leaving network	When this bit is set to one, it indicates the sender of the hello command frame is leaving the network.
b_6	Addresses of multicast groups not presented	When this bit is set to one, it indicates the addresses of multicast groups field is empty and does not need to be process.
b_5	Newly joined multicast group	When this bit is set to one, it indicates the addresses of multicast groups field only includes those multicast groups of which this device just became a member since last hello command frame.
b_4	Newly left multicast group	When this bit is set to one, it indicates the addresses of multicast groups field only includes those multicast groups from which this device just left since last hello command frame.
b_3	Full multicast membership update	When this bit is set to one, it indicates the addresses of multicast groups field includes a full list of multicast groups of which this device is a member.
$b_{2–0}$	Reserved	–

The Number of One-hop Neighbors field should contain the number of neighbors that have been assigned logical addresses and are one hop away from the source. This information can be obtained from the neighbor list as defined in Table 46.

The Number of Multicast Groups field should contain the number of the multicast group address to be included in the Addresses of Multicast Groups field.

The Addresses of One-hop Neighbors field should contain the short addresses of all the devices that have been assigned logical addresses and are one hop away from the source.

The Addresses of Multicast Groups field should contain the addresses of muliticast groups this device has just joined, left or is a member of.

5.3.2.2.4 Neighbor information request command

The neighbor information request command is used for a mesh device to request neighbor information from neighbors from which expected hello commands are not received. The payload of the neighbor information request frame should be formatted as illustrated in Figure 13.

Octets: 1	1	1	Variable
Command Frame Identifier	TTL	Number of Extended Neighbors	Addresses of Extended Neighbors
Data Frame Mesh Sublayer Payload			

Figure 13—Neighbor information request command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have a value of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be broadcast and acknowledgement is not requested.

The Destination Address field should contain the logical broadcast address. The Source Address field should contain the logical address of the source.

The TTL field should always be set to one to indicate the frame is to broadcast to one-hop neighbors only.

The Number of Extended Neighbors field should contain the number of extended neighbors for which information is requested.

The Addresses of Neighbors field should contain the logical addresses of all extended neighbors for which information is requested.

5.3.2.2.5 Neighbor information reply command

The neighbor information reply command is sent in response to received neighbor information request command. The payload of the command frame should be formatted as illustrated in Figure 14.

Octets: 1	**1**	**Variable**
Command Frame Identifier	Number of Extended Neighbors	List of Extended Neighbors
Data Frame Mesh Sublayer Payload		

Figure 14— Neighbor information reply command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is not requested.

The Destination Address field should contain the logical address of the destination. The Source Address field should contain the logical address of the source.

The Number of Extended Neighbors field should contain the number of extended neighbors that are listed in the corresponding neighbor information request frame for which this device can provide information.

The List of Extended Neighbors field should contain a list of extended neighbors for which information is provided. The format of each entry in the List of Extended Neighbors field should be formatted as illustrated in Figure 15. The Beginning Address, End Address and Tree Level fields are defined in neighbor list as illustrated in Table 46.

Octets: 2	**2**	**1**
Beginning Address	End Address	Tree Level

Figure 15— Format of each entry in the List of Extended Neighbors field

5.3.2.2.6 Link state command

The link state command is used for a mesh device to inform its neighbors that link state mismatch may exist among neighboring devices. The payload of the command frame should be formatted as illustrated in Figure 16.

Octets: 1	**1**	**Variable**	**Variable**
Command Frame Identifier	Number of Neighbor	Neighbor Addresses	Connectivity Bitmap
Data Frame Mesh Sublayer Payload			

Figure 16—Link state command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address field should contain the logical address of the destination. The Source Address field should contain the logical address of the source.

The Number of Neighbor field should contain the number of (*meshTTLOfHello*-1)-hop neighbors.

The Neighbor Addresses field should contain a list of all the (*meshTTLOfHello*-1)-hop neighbors that are being reported by this frame. Each neighbor address in the list is of 2 octets in length. Note the sequence of each neighbor in the list is important for reconstructing a connectivity matrix at the receiver side.

The Connectivity Bitmap field should contain a subset of the connectivity matrix as illustrated in Table 47. This subset should include the (*meshTTLOfHello*-1)-hop connectivity matrix in the table. In the Connectivity Bitmap, a bit is set to one when there is a connection corresponding to two devices in the connectivity matrix (the cross point in the matrix); otherwise, it is set to zero. To generate the Connectivity Bitmap for *n* (*meshTTLOfHello*-1)-hop neighbors, starting from *me* (first row, first column) to nb_n (first row, last column), the bitmap of the first row of the connectivity matrix is turned into CEILING [(n+1)/8] octets (rounded up to the nearest integer), where the last octet may not be full and should be padded with all zeros (the padded bits won't be read anyway). For example, when *n* equals to ten, the first row will be stored in two octets with the last 6 bits of the last octet equal to zeros. This process continues until all *(n+1)* rows are converted into an octet array. The total length of this octet array equals to CEILING [(n+1)/8]×(n+1).

5.3.2.2.7 Link state mismatch command

The link state mismatch command is used for a mesh device to inform its neighbors that link state mismatch has happened and needs to be corrected. The payload of the command frame should be formatted as illustrated in Figure 17.

Octets: 1	**1**	**1**	**Variable**
Command Frame Identifier	TTL	Number of Neighbors	Addresses of Neighbors
Data Frame Mesh Sublayer Payload			

Figure 17— Link state mismatch command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be broad and acknowledgement is not requested.

The Destination Address field should contain the logical broadcast address. The Source Address field should contain the logical address of the source.

The TTL field should contain the number of hops the link state mismatch message is allowed to propagate, which could be up to *meshExtendedNeighborHopDistance*-1 (caused by neighbors at most *meshExtendedNeighborHopDistance* –1 hops away).

The Number of Neighbors field should contain the number of neighbors that are related to the link state mismatch.

The Addresses of Neighbors field should contain the addresses of neighbors that are related to the link state mismatch. The neighbors that cause link state mismatch could be *meshExtendedNeighborHopDistance* –1 hops away at most and the value of TTL should be equal to the hop number of the farthest neighbor that causes the link state mismatch.

5.3.2.2.8 Probe command

The probe command is used for a mesh device to test the connection to a specific neighbor. The payload of the command frame should be formatted as illustrated in Figure 18.

Octets: 1
Command Frame Identifier
Data Frame Mesh Sublayer Payload

Figure 18— Probe command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address field should contain the logical address of the destination. The Source Address field should contain the logical address of the source.

5.3.2.2.9 G-JREQ command

The G-JREQ command frame allows a device to join a specific multicast group. The payload of the frame should be formatted as illustrated in Figure 19.

Octets: 1	2	1	1
Command Frame Identifier	Group Address	Join Options	Hop Count
Data Frame Mesh Sublayer Payload			

Figure 19— G-JREQ command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have a value of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address could be the logical address of a mesh device, the MC, or the group coordinator (GC). In the case the destination address is the logical address of the MC (b_6 of the join option field is set to one) or the GC (b_5 of the join option field is set to one), the destination address is not changed until the MC or the GC is reached. In the case the destination address is neither the MC nor the GC, it will be replaced at an intermediate device by the address of the next hop calculated by the joining algorithm described in 5.5.8.2.1.

The Source Address should always be set to the frame originator's unicast address and kept unchanged until the frame reaches a device that can respond with a join reply.

The Group Address should be set to the multicast group address passed from the next higher layer.

The Join Options field is described in Table 37.

The Hop Count field is an integer and it records the hop distance this G-JREQ command has traveled.

Table 37— Join options field of G-JREQ frame

Bit	Options	Description
b_7	Join as GC	This bit is set to one if the source address field in the mesh sublayer header contains the GC's logical address. It indicates to the MC that this device will be the GC for the multicast group. In all other cases, this bit is set to zero.
B_6	MC as destination	This bit is set to one if the destination address field in the mesh sublayer header contains the MC's logical address. It indicates this G-JREQ command is sent to the MC. It should be set to zero if the G-JREQ is not sent to the MC.
B_5	GC as destination	This bit is set to one if the destination address field in the mesh sublayer header contains the GC's logical address. It indicates this G-JREQ command is sent to the GC. It should be set to zero if the G-JREQ is not sent to the GC.

Table 37—Join options field of G-JREQ frame *(continued)*

Bit	Options	Description
B_4	G-JREP requested	This bit is set to one if a G-JREP command frame is requested when the G-JREQ command reaches the destination. It is set to zero otherwise.
B_3	Act as router	When this bit is set to one the intermediate devices which receive this G-JREQ should change their status to Router (if they are not group members (GMs) or the GC) without waiting for a G-JREP frame.
B_2	End device	This bit is set to one if the originator of this command frame is an end device; otherwise, it is set to zero.
b_1–b_0	Reserved	–

5.3.2.2.10 G-JREP command

The G-JREP command frame is composed in response to a G-JREQ command frame and sent by a mesh device, the GC or the MC that can connect the requestor to the multicast group. The payload of the command frame should be formatted as illustrated in Figure 20.

Octets: 1	2	1	1
Command Frame Identifier	Group Address	Join Options	Status
Data Frame Mesh Sublayer Payload			

Figure 20—G-JREP frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address should always be set to the G-JREQ originator's unicast address and not to be changed by intermediate devices. The Source Address should always be set to the frame originator's unicast address and kept unchanged until the frame reaches the final destination.

The Group Address should be set to the multicast group address received from G-JREQ frame.

The Join Options field is illustrated in Table 38.

The Status field provides the status of the group join process as described in Table 39.

Table 38—Format of Join Option field

Bit	Options	Description
b_7	ActAsRouter	When this bit is set to one the intermediate devices which receive this G-JREP should change their status to Router (if they are not GMs or the GC).
b_6–b_0	Reserved	–

Table 39—Format of Status field

Bit	Status	Description
b_7	SUCCESS	This bit is set to one if the joining device has successfully joined the targeted device (a GM, the MC, or the GC).
b_6	FAILURE	This bit is set to one if the joining device has failed to join the targeted device. The reason can be given by other bits of this field.
b_5	GC_NOT_YET_JOINED	This bit is set to one if the G-JREQ has reached the MC but the GC has not joined yet.
b_4-b_0	Reserved	–

5.3.2.2.11 G-LREQ command

The G-LREQ command frame allows an end device to leave a specific multicast group. The payload of the frame should be formatted as illustrated in Figure 21.

Octets: 1	2
Command Frame Identifier	Group Address
Data Frame Mesh Sublayer Payload	

Figure 21—G-LREQ command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have a value of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address could be set to the logical address of parent device.

The Source Address should always be set to the frame originator's logical address.

The Group Address should be set to the multicast group address passed from the next higher layer.

5.3.2.2.12 Group leave reply command

The group leave reply command frame is composed by a mesh device, the GC or the MC and sent to one of its child device that is an end device. The payload of the frame should be formatted as illustrated in Figure 22.

Octets: 1	**2**
Command Frame Identifier	Group Address
Data Frame Mesh Sublayer Payload	

Figure 22—Group leave reply frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address should always be set to the G-LREQ originators logical. The Source Address should always be set to the frame originator's logical address.

The Group Address should be set to the multicast group address received from G-LREQ frame.

5.3.2.2.13 Wakeup notification command

The wakeup notification command frame allows a mesh device to notify its own schedule information to neighbors in the optional energy saving mode. When meshASESOn is set to TRUE, it should be transmitted at the beginning of the active duration for every wakeup interval. The payload of the frame should be formatted as illustrated in Figure 23.

Octets: 1	**1**
Command Frame Identifier	Asynchronous Energy Saving (ASES) Time Info
Data Frame Mesh Sublayer Payload	

Figure 23—Wakeup notification command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be a non-reliable broadcast.

The Destination Address should be set to 0xffff. The Source Address should be set to the mesh address of the source device.

ASES Time Info should contain *meshWakeupOrder* and *meshActiveOrder* of the mesh device. These variables should be set as illustrated in Figure 24.

b_{7-4}	b_{3-0}
Wakeup Order	Active Order

Figure 24—Format of ASES Time Info field

5.3.2.2.14 Extension request command

The extension request command frame allows a mesh device to request an extension of the active duration at the destination for.the duration specified in the extension time field of the mesh sublayer payload. The payload of the frame should be formatted as illustrated in Figure 25.

By MAC layer broadcasting, the mesh sublayer detects channel activity without filtering at the MAC layer. If the destination is matched (including the broadcast address); a receiving device should extend the active time by Extension Time in the payload. If the destination addres is not matched, a device should ignore the request.

Octets: 1	**1**
Command Frame Identifier	Extension Time
Data Frame Mesh Sublayer Payload	

Figure 25— Extension request command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be a non-reliable broadcast.

The Destination Address should be set to the mesh address of the destination device or 0xffff. The Source Address should always be set to the mesh address of the source device.

The Extension Time contains required amount of time for extending the active time in the unit of *meshcTimeUnit.*

5.3.2.2.15 Extension Reply command

The extension reply command frame allows a mesh device to confirm that the extension request of the active duration is accepted. The payload of the frame should be formatted as illustrated in Figure 26.

By MAC layer broadcasting, the mesh layer detects channel activity without filtering at the MAC layer. If the destination is not matched, a node should ignore the reply.

Octets: 1
Command Frame Identifier
Data Frame Mesh Sublayer Payload

Figure 26—Extension reply command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be a non-reliable broadcast.

The Destination Address field should always be set to the mesh address of the device that sent the EREQ. The Source Address field should always be set to the mesh address of the source.

5.3.2.2.16 Synchronization request command

The synchronization request command frame allows a mesh device to synchronize with all its child devices. The payload of the frame should be formatted as illustrated in Figure 27.

Octets: 1	4	3
Command Frame Identifier	Timestamp Value	Synchronization Specification
Data Frame Mesh Sublayer Payload		

Figure 27—Synchronization request command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfield should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be a non-reliable broadcast.

The Destination Address field should be set to 0xffff. The Source Address field should be set to the mesh address of the source device.

The Timestamp Value field should contain the previous transmitted time of the first synchronization request frame. This value is obtained when MCPS-DATA.confirm is received. The Synchronization Specification field should be formatted as illustrated in Figure 28.

b_{23-20}	b_{19-16}	b_{15}	b_{14-8}	b_{7-4}	b_{3-0}
Hop Count	Synchronization Region	Synchronization Frame Order	Synchronization Interval	WO	AO

Figure 28—Format of Synchronization Specification field

The Hop Count subfield should contain the number of hops and be increased by one for each request frame. The Synchronization Region subfield should contain the number of hops to be synchronized by a mesh coordinator or region synchronizer during synchronization duration. The Synchronization Frame Order subfield should contain the order of synchronization request frame. If the value of the Synchronization Frame Order is zero, it indicates the first synchronization request frame. If the value of the Synchronization Frame Order is one, it indicates the second synchronization request frame. The Synchronization Interval subfield should contain the number of wakeup interval times for periodic synchronization. The WO subfield should contain the value of wakeup interval. The AO subfield should contain the length of time during which the time structure is active.

5.3.2.2.17 Synchronization reply command

The synchronization reply command allows a mesh device to confirm its parent device that the synchronization request command frame has been received. The payload of the frame should be formatted as illustrated in Figure 29.

Octets: 1
Command Frame Identifier
Data Frame Mesh Sublayer Payload

Figure 29—Synchronization reply command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfield should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address field should always be set to the mesh address of the device that sent the synchronization request command frame. The source address field should be set to the mesh address of the source device.

5.3.2.2.18 Reservation request command

The reservation request command allows a mesh device to request a reservation of data transmission in the specific reservation slot period of SES time structure. The payload of the frame should be formatted as illustrated in Figure 30.

Octets: 1	2	2	2	1
Command Frame Identifier	Previous Address	Next Address	End Address	Reservation Slot Number
Data Frame Mesh Sublayer Payload				

Figure 30—Reservation request command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfield should have values of one indicating the 16-bit short addresses and the Transmission Options field should have corresponding values, indicating this frame should be a non-reliable broadcast.

The Destination Address field should be set to mesh sublayer broadcast address. The source address field should be set to the mesh address of the source device.

The Previous Address field should be set to the mesh address of a device that previously sent reservation request command.

The Next Address field should be set to the mesh address of a device that should be received this reservation request frame.

The End Address field should be set to the mesh address of final destination device.

The Reservation Slot Number field should contain the order of data transmission in inactive duration of SES time structure and is increased by one for each request frame.

5.3.2.2.19 Reservation reply command

The reservation reply command allows a mesh device to confirm its parent device that the reservation request command frame has been received. The payload of the frame should be formatted as illustrated in Figure 31.

Octets: 1
Command Frame Identifier
Data Frame Mesh Sublayer Payload

Figure 31— Reservation reply command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfield should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgement is requested.

The Destination Address field should always be set to the mesh address of the device that sent the reservation request command frame. The Source Address field should be set to the mesh address of the source device.

5.3.2.2.20 Rejoin notification command

The rejoin notification command is used to inform the relevant devices of the movement of the portable device as described in 5.5.11.3. The payload of the frame should be formatted as illustrated in Figure 32.

Octets: 1	**1**	**8**	**2**	**2**
Command Frame Identifier	TTL	IEEE Address	Previous Short Address	New Short Address
Data Frame Mesh Sublayer Payload				

Figure 32— Rejoin notification command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfield should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating whether this frame should be unicast or broadcast, and whether the acknowledgement is requested.

If the frame is to be sent to the old parent, the Destination Address field should be set to the old parent's address, the TTL field should be set to zero, and unicast is used; if the frame is to be sent to all neighbors, the Destination Address should be set to 0xffff and the TTL is set to the broadcast radius. The Source Address field should contain the MAC address of the source. The details are can be found in 5.5.11.3.

The IEEE Address field should contain the 64-bit extended address of the source.

The Previous Short Address field should contain the short address assigned to the source device before the movement.

The New Short Address field should contain the short address assigned to the source device by the new parent device after the movement.

5.3.2.2.21 Traceroute request command

The traceroute request command is used to monitor the routing path between a source and a destination device. The payload of the frame should be formatted as illustrated in Figure 33.

Octets: 1	**1**	**1**
Command Frame Identifier	TTL	Sequence Number
Data Frame Mesh Sublayer Payload		

Figure 33— Traceroute request command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgment is requested.

The Destination Address should be set to the logical address of the destination. The Source Address field should contain the logical address of the source.

The TTL field should contain the number of hops the traceroute request is allowed to propagate. This information is obtained from the MHME-TRACE-ROUTE.request primitive defined in 5.2.2.21.

The Sequence Number field is an integer generated and maintained by the sender of the trace route request frame and increased by one for each request frame.

5.3.2.2.22 Traceroute reply command

The traceroute reply command is used to respond to the traceroute request command. The payload of the frame should be formatted as illustrated in Figure 34.

Octets: 1	**1**
Command Frame Identifier	Sequence Number
Data Frame Mesh Sublayer Payload	

Figure 34— Traceroute reply command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgment is requested.

The Destination Address should be set to the logical address of the source of the traceroute request command. The Source Address field should contain the logical address of the destination of the traceroute request command.

The value of the Sequence Number field should be taken from the same field of the trace route request frame.

5.3.2.2.23 Leave command

The leave command is used by a parent device to request a child device to leave the mesh network. The payload of the frame should be formatted as illustrated in Figure 35.

Octets: 1	**1**
Command Frame Identifier	Leave Control
Data Frame Mesh Sublayer Payload	

Figure 35—Leave command frame payload format

In the Frame Control field, both Destination and Source Address Mode subfields should have values of one indicating the 16-bit short addresses and the Transmission Options subfield should have corresponding values, indicating this frame should be unicast and acknowledgment is requested.

The Destination Address should be set to the logical address of the destination. The Source Address field should contain the logical address of the source.

The Leave Control field should be formatted as shown in Table 36. The receiver of the leave command is requested to remove all its child devices if the RemoveChildren bit is set to one.

b_7	b_{6-0}
RemoveChildren	Reserved

Figure 36—Format of Leave Control field

5.3.3 Mesh sublayer information in MAC beacon payload

This subclause describes mesh sublayer information carried by the MAC layer beacon payload. The information provides detailed descriptions of existing mesh networks and the devices that send out the beacons. It helps new devices to discover the existing mesh networks and to select the right parent device to join, as described in 5.2.2 and 5.5.2. It also provides necessary parameters to support functions such as low power router and portiblility. The bit sequence of the beacon payload should be formatted as in Figure 37.

Bits: 0–3	**4–11**	**12**	**13**	**14**	**15**	**16**	**17–20**	**21–24**
Mesh Version	Tree Level	Accept Mesh Device	Accept End Device	Reliable Broadcast	Sync Energy Saving	Async Energy Saving	Active Order	Wakeup Order

Figure 37—Format of MAC Layer beacon payload

The details of the mesh sublayer information fields need to be included in the MAC beacon payload are described in Table 40.

Table 40—Mesh sublayer information field in beacon payload

Name	Valid range	Description
MeshVersion	0x00–0x0f	The version of the mesh sublayer protocol in use in the discovered network.
TreeLevel	0x00–0xff	The tree level of the beacon sender.
AcceptMeshDevice	one or zero	This field indicates whether the mesh network accepts new mesh devices. A value of one indicates the device is capable of accepting join requests from other mesh devices; the value should be set to zero otherwise.
AcceptEndDevice	one or zero	This field indicates whether the mesh network accepts new end devices. A value of one indicates the device is capable of accepting join requests from end devices; the value should be set to zero otherwise.
ReliableBroadcast	one or zero	This field indicates whether the optional reliable broadcast is supported. A value of one indicates the device is capable of supporting reliable broadcast; the value should be set to zero otherwise.
SyncEnergySaving	one or zero	This field indicates whether the optional synchronous energy saving feature is supported. A value of one indicates the device is capable of supporting synchronous energy saving; the value should be set to zero otherwise.
AsyncEnergySaving	one or zero	This field indicates whether the optional asynchronous energy saving feature is supported. A value of one indicates the device is capable of supporting asynchronous energy saving; the value should be set to zero otherwise.
ActiveOrder	0x00–0x0f	This field defines the length of the active duration within the wakeup interval. It is to be read only when either the SyncEnergySaving or the AsyncEnergySaving field is TRUE.
WakeupOrder	0x00–0x0f	This field defines the interval of the periodical wakeup. It is to be read only when either the SyncEnergySaving or the AsyncEnergySaving field is TRUE.

For the mesh coordinator, the beacon payload is constructed immediately after a new mesh network is formed. For all other devices, it is constructed immediatedly after the devices have joined the network, or after any parameters in Table 40 have changed. The constructed beacon payload is sent to the MAC layer PIB using the MLME-SET.request primitive. The MAC layer PIB attribute for storing beacon payload is

macBeaconPayload. The MAC PIB attribute *macBeaconPayloadLength* should be set to the length of the *macBeaconPayload* attribute.

5.4 Mesh sublayer constants, mesh information base and statue values

5.4.1 Mesh sublayer constants

Mesh sublayer constants defines the characteristics of the mesh sublayer. They are presented in Table 41.

Table 41—Mesh sublayer constants

Constants	Type	Value	Description
meshcCoordinatorCapability	Boolean	TRUE, FALSE	This constant indicates whether the device is capable of acting as a mesh coordinator (MC). It is set at device build time. TRUE indicates it is capable of being an MC while FALSE indicates the device is not capable of being an MC.
meshcBroadcastAddress	Integer	0xffff	The short logical broadcast address at mesh sublayer.
meshcMaxMeshHeaderLength	Integer	0x12	The maximum length in octets of the mesh header (18 bytes).
meshcTimeUnit	Integer	1	Time unit for mesh sublayer power saving. The unit is millisecond.
meshcBaseActiveDuration	Integer	*meshcTimeUnit*× 5	The time basis of the active duration when the active order is equal to zero.
meshcMaxLostSynchronization	Integer	3	The number of consecutive synchronization failure.
meshcReservationSlotDuration	Integer	625	The time period of reserved data transmission in inactive duration of SES time structure (in MAC sublayer symbol periods).

5.4.2 Mesh information base

The mesh information base contains variables that can be tuned to allow the mesh sublayer protocol to meet the requirements of different application scenarios. The variables can be read and written from the next higher layer through the mesh sublayer services defined in 5.2. The mesh information base is illustrated in Table 42.

Table 42—Mesh information base

Attribute	ID	Type	Range	Description	Default Value
meshNbOfChildren	0xa1	Integer	0x00–0xff	The number of devices that have joined this device.	0x00
meshCapabilityInformation	0xa2	Bit Vector	As described in Table 16	This field indicates the capability of the device.	0x00
meshTTLOfHello	0xa3	Integer	0x00–0xff	The number of hops the hello command frames are allowed to travel.	0x01
meshTreeLevel	0xa4	Integer	0x00–0xff	The number of hops this device is from the Mesh coordinator through its parent.	0x00
meshPANId	0xa5	Integer	0x0000–0xffff	The PAN ID of this mesh network. It should have the same value of the *macPANId*.	0xffff
meshNeighborList	0xa6	Set	Variable	This attribute contains the information of neighbors of the device as defined Table 46.	Null set
meshDeviceType	0xa7	Type Enumeration	MESH_ DEVICE, END_ DEVICE	This attribute is set to END_DEVICE if the device is an End Device, MESH_DEVICE if the device is a Mesh Device.	END_DEVICE
meshSequenceNumber	0xa8	Integer	0x00–0xff	A sequence number used to identify outgoing frames.	Random value from within the range
meshNetworkAddress	0xa9	Integer	0x0000–0xfffe	The mesh sublayer address of this device. This attribute reflects the value of the MAC PIB attribute *macShortAddress* as defined in IEEE Std 802.15.4-2006 and any changes made by the higher layer will be reflected in the MAC PIB attribute value as well.	0xffff
meshGroupCommTable	0xaa	Set	See Table 48	This table records all multicast addresses of which this device is a member and its status in the group.	Null Set

Table 42— Mesh information base *(continued)*

Attribute	ID	Type	Range	Description	Default Value
meshAddressMapping	0xab	Set	variable, as defined in Table 43	This attribute maps 64-bit IEEE addresses to 16-bit short address.	Null Set
meshAcceptMeshDevice	0xac	Boolean	TRUE or FALSE	This attribute indicates whether this device allows other mesh devices to join as its children.	FALSE
meshAcceptEndDevice	0xad	Boolean	TRUE or FALSE	This attribute indicates whether this device allows other end devices to join as its children.	FALSE
meshChildNbReportTime	0xae	Integer	0x0000–0xffff	This attribute indicates the time in seconds to start reporting the number of child devices after this device has successfully joined the network.	Set at build time (the value of this attribute may vary greatly for different applications)
meshProbeInterval	0xaf	Integer	0x0000–0xffff	This attribute sets the probe interval in seconds for a neighbor if its status in the neighbor list is unknown.	0x10
meshMaxProbeNum	0xb0	Integer	0x00–0xff	The maximum number an unknown neighbor is going to be probed.	0xff
meshMaxProbeInterval	0xb1	Integer	0x0000–0xffff	This attribute sets the maximum probe interval in seconds for a neighbor if its status in the neighbor list is unknown.	0xffff
MaxMulticastJoinAttempts	0xb2	Integer	0x00–0xff	The maximum number of attempts a device should try to join a multicast group before considering the joining process a failure.	0x07

Table 42— Mesh information base *(continued)*

Attribute	ID	Type	Range	Description	Default Value
meshRBCastTXTimer	0xb3	Integer	0x0000–0xffff	This attribute indicates the time in seconds that a broadcasting device needs to wait before rebroadcasting the data frame.	Set at build time (the value of this attribute may vary greatly for different applications)
meshRBCastRXTimer	0xb4	Integer	0x0000–0xffff	This attribute indicates the maximum waiting time in seconds that the receiving device forwards the broadcast data frame.	Set at build time (the value of this attribute may vary greatly for different applications)
meshMaxRBCastTrials	0xb5	Integer	0x00–0xff	This attribute indicates the maximum number of broadcasting trials before issuing MHME-LEAVE.indication primitive to avoid possible unreachability of one or more neighbor devices.	Set at build time (the value of this attribute may vary greatly for different applications)
meshASESOn	0xb6	Boolean	TRUE or FALSE	This attribute indicates whether this device is in ASES mode.	FALSE
meshASESExpected	0xb7	Boolean	TRUE or FALSE	This attribute indicates whether this device is capable of ASES.	FALSE
meshWakeupOrder	0xb8	Integer	0–15	This specifies how often a device transmits its wakeup notification. If the wakeup order, *WO* = 15, the device will not transmit a periodic wakeup notification.	15
meshActiveOrder	0xb9	Integer	0–15	The length of the active duration within a wakeup interval, including the wakeup notification. If *WO* = 15, this value is ignored.	15

Table 42— Mesh information base *(continued)*

Attribute	ID	Type	Range	Description	Default Value
meshDestActiveOrder	0xba	Integer	0–15	The length of the active duration of the destination device. When *meshASESOn* is FALSE, this value is ignored.	0
meshEREQTime	0xbb	Integer	0–255	The maximum number of meshcTimeUnits to wait for the next EREQ or a data frame following an EREQ.	30
meshEREPTime	0xbc	Integer	0–255	The maximum number of meshcTimeUnits to wait for an EREP after transmitting EREQ.	15
meshDataTime	0xbd	Integer	0–255	The maximum number of meshcTimeUnits to wait for the EREQ or a Data frame following a data frame.	15
meshMaxNumASESRetries	0xbe	Integer	0–15	The maximum number of retransmissions in ASES.	2
meshSESOn	0xbf	Boolean	TRUE or FALSE	This attribute indicates whether this node is in SES mode.	FALSE
meshSESExpected	0xc0	Boolean	TRUE or FALSE	This attribute indicates whether this node is capable of SES.	FALSE
meshSyncInterval	0xc1	Integer	0x00–0xff	The number of wakeup interval times for periodic synchronization.	0x0a
meshMaxSyncRequestAttempts	0xc2	Integer	0x00–0xff	The maximum number of times a node should try to transmit a sync request frame before considering the sync procedure failure.	0x03
meshSyncReplyWaitTime	0xc3	Integer	0–255	The maximum number of meshTimeUnits to wait for synchronization reply frame after transmitting synchronization request frame.	50
meshFirstTxSyncTime	0xc4	Integer	0x00000000–0xffffffff	The time that the device transmitted its first synchronization request frame, in MAC symbol periods.	0x00000000

Table 42— Mesh information base *(continued)*

Attribute	ID	Type	Range	Description	Default Value
meshFirstRxSyncTime	0xc5	Integer	0x00000000–0xffffffff	The time that the device received its first synchronization request frame, in MAC symbol periods.	0x00000000
meshSecondRxSyncTime	0xc6	Integer	0x00000000–0xffffffff	The time that the device received its second synchronization request frame, in MAC symbol periods.	0x00000000
meshRegionSynchronizaerOn		Boolean	TRUE or FALSE	This attribute indicates whether this node is a region synchronizer.	FALSE
meshExtendedNeighborHopDistance	0xc7	Integer	0x00–0xff	The predetermined hop distance between a device and its extended neighbors.	0x03
meshRejoinTimer	0xc8	Integer	0x0000–0xffff	The time between a child device leaves the network and the child device is deleted from the neighbor list.	0xffff

Table 43—Mesh address mapping

64-bit IEEE Address	16-bit Short Address
A valid 64-bit IEEE Address	A valid 16-bit short address (0x0000–0xfffe)

5.4.3 Mesh status values

This subclause defines status values used by confirmation primitives that reports on the status of the corresponding request primitives. Values for mesh sublayer status parameters appear in Table 44.

Table 44—Mesh status values

Name	Value	Description
–	0x01–0xb0	Reserved
INVALID_REQUEST	0xb1	A requested service was invalid.
NOT_PERMITTED	0xb2	Join was failed because it was not permitted.
NO_NETWORKS	0xb3	No network was found.
READ_ONLY	0xb4	The attribute to be set is a read-only attribute.
RECEIVE_SYNC_LOSS	0xb5	A synchronization loss notify frame was received.
RECEIVE_SYNC_RESPONSE	0xb6	A synchronization response frame was received.
STARTUP_FAILURE	0xb7	A network could not be started because of no suitable channel or PAN identifier.
SYNC_FAILURE	0xb8	Synchronization request was not successful because at least one synchronization response frame was not received.
SYNC_LOSS	0xb9	A synchronization request frame was not received at the scheduled time.
SYNC_SUCCESS	0xba	SYNC request was successful.
TRACEROUTE_TIMEOUT	0xbb	Traceroute reply command frame was not arrived within the requested response time.
TRACEROUTE_UNREACHABLE	0xbc	Traceroute failed because the destination device is unreachable.
UNKNOWN_CHILD_DEVICE	0xbd	A device requested to leave is not a child device.

5.5 Mesh function description

This clause describes the function the mesh sublayer provides to the next higher layer. These functions include unicast and multicast addressing, unicast routing algorithm, and multicast routing algorithm.

5.5.1 Starting a mesh network

A mesh network usually starts first by the Mesh coordinator turning on its power. The next higher layer of an MC that is ready to start its own mesh network should issue a MHME-START-NETWORK.request, as defined in 5.2.2.3, to the mesh sublayer. The mesh sublayer will then perform two MAC layer scans, first

an energy detection scan and then an active scan. This is performed by issuing the MLME-SCAN.request primitive to the MAC sublayer.

When the MLME-SCAN.confirm primitive comes back from the MAC sublayer, the mesh sublayer of the MC will gather a list of networks operating in its neighborhood. If the list is not empty, the MC should record the neighbor devices from which it received the signals in the Mesh Discovery Table as defined in 5.2.2.1. The useful information includes PAN ID, address mode, and address of each device (both 16-bit and 64-bit addresses are possible), the channel each neighbor is currently operating on, superframe related information, link quality from each neighbor, and any other information the implementers deem necessary to be stored for future usage.

After gathering the information from existing networks, the mesh sublayer of the MC will decide the channel and PAN ID it wants to use for its network. The MC can now start the network by issuing the MLME-START.request primitive with the PANCoordinator parameter set to TRUE to the MAC sublayer. The BeaconOrder, SuperframeOrder and other parameters of the MLME-START.request primitive will be the same as those given to the MHME-START-NETWORK.request.

Upon receiving the MLME-START.request primitive, the MAC sublayer enters the operating mode and is ready to transmit its beacons when requested to allow new mesh devices to join the mesh network.

The detailed information on starting a new mesh network is described in 5.2.2.3.

5.5.2 Joining a mesh network

Joining a mesh network usually contains the following steps:

— Discovering existing channels/networks;

— Selecting a channel/network and parent device to join;

The first thing a non-MC device should do is to detect whether there are WPAN networks operating in its vicinity. The next higher layer of the device will issue a MHME-DISCOVER.request, as defined in 5.2.2.1, to its mesh sublayer. The mesh sublayer will attempt to discover networks operating within the device's operating area by performing an active scan over the channels specified in the MHME-DISCOVER.request primitive. This scan is performed by issuing the MLME-SCAN.request primitive to the MAC layer of the device.

When the MLME-SCAN.confirm primitive comes back from the MAC layer, the mesh sublayer of the device will gather a list of networks operating in its neighborhood. If the list is not empty, the MC should record the neighbor devices from which it received the signals in the mesh discovery table as described in 5.2.2.1. The useful information that should be recorded includes PAN ID, address mode and address of each device (both 16-bit and 64-bit addresses are possible), the channel each neighbor is currently operating on, superframe related information, link quality from each neighbor, etc., as defined in IEEE Std 802.15.4-2006. The mesh sublayer will then issue an MHME-DISCOVER.confirm primitive containing the information about the discovered networks with a Status parameter value equal to that returned with the MLME-SCAN.confirm to the next higher layer.

Upon receiving the MHME-DISCOVER.confirm primitive, the next higher layer is able to start the joining process. This is done by the AME issuing an MHME-JOIN.request to the MHME. The AME should specify in the primitive the parameters such as the PAN ID to join, the process that should be used to join in the MAC layer, the type of device it will be operating as (mesh device or end device), etc. Note that the channel-related parameters only apply to the case when the device is trying to rejoin the network. Based on the information given in the MHME-JOIN.request, the mesh sublayer will try to find the best neighbor device to join, if multiple candidates are available. The criteria needing to be considered include tree level

and link quality. The implementers can take other factors into account if their application has special requirements.

After determining the neighbor device to join, the device issues an MLME-ASSOCIATE.request primitive to its MAC layer with the parameters properly set, as described in 5.2.2.7. Successful or not, the mesh sublayer will receive the MLME-ASSOCIATE.confirm from the MAC layer indicating the status of the association process. If the device has successfully joined the parent it has chosen, it should update its database entries related to this parent, e.g. record its parent in its neighbor list and specify the relationship as "parent". The mesh sublayer should also send a MHME-JOIN.confirm with the Status parameter properly set to its next higher layer.

If the device joins the network as a mesh device, its AME may initiate the mesh functions the device is able to perform after it has joined the network. The major mesh functions include relaying unicast, multicast and broadcast data frames for other devices and allowing other devices to join the network through it. The AME does this by issuing an MHME-START-DEVICE.request to the mesh sublayer. The mesh sublayer will, in turn, issue the MLME-START.request primitive to the MAC layer to start the device as a mesh device. If the received MLME-START.confirm primitive returns a positive value, it means the device has successfully become a mesh device and is able to carry out the functions associated with a mesh device. The mesh sublayer should issue a MHME-START-DEVICE.confirm primitive to AME no matter the results of the MLME-START.confirm primitive.

If the device joins the network as an end device, the device can start performing its task after transmitting the MHME-JOIN.confirm primitive to AME.

The details of the joining and initiation process can be found in 5.2.2.1, 5.2.2.5, and 5.2.2.7.

5.5.3 Address assignment

By binding logical addresses to the network topology, routing can be carried out without going through route discovery. Address assignment is broken down into two stages: association as described in 5.5.2 and 5.5.3.1, and address assigning as described in 5.5.3.2.

5.5.3.1 Association

During the association stage, beginning from the root, devices gradually join the network and a tree is formed. But this tree is not a logical tree yet, since no device has been assigned an address. There is no mesh-level limitation on the number of children a device can have. A device should determine by itself how many devices (therefore, how many branches) it should accept according to its capability and other factors. Note in this stage, the network is not functional, i.e. no data can be transferred from one device to the other.

5.5.3.2 Address assigning

After a branch reaches its bottom, that is, no more devices wait for joining the network (a suitable timer, *meshChildNbReportTime*, can be used for this purpose), a bottom-up procedure shall be used to calculate the number of devices along each branch, as shown in Figure 38. The numbers in square brackets indicate the numbers of devices within branches below a certain device.

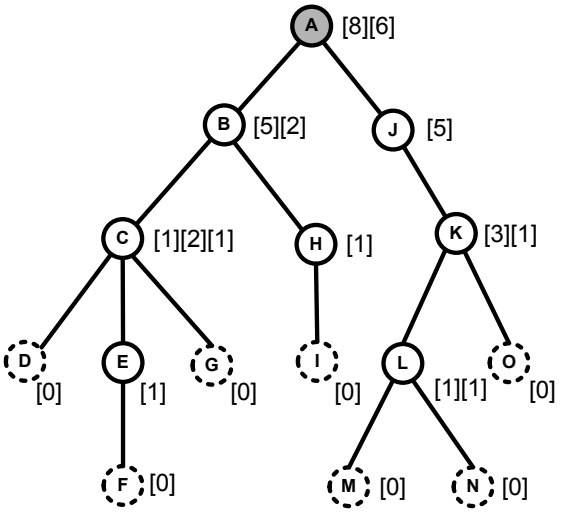

Figure 38—Calculation of number of devices along each branch

To report the number of children, each device shall proceed as follows. Whenever a device joins the network, it should start a timer. The duration of the timer is defined by the MeshIB attribute *meshChildNbReportTime* as described in Table 42. The device becomes a leaf device if no other devices join it before the timer expires. A leaf device should immediately send a children number report frame to its parent as described in 5.3.2.2.1, setting the number of children field to one, and the number of requested addresses field to a value equal to one, or larger than one if it wants to reserve some addresses for possible future use. When a non-leaf device receives a children number report frame, it shall record the number of children and the number of requested addresses for that branch, and then check whether each of its children has reported the number of children. If each of its children has reported the number of children, it shall report the number of children to its parent. The number of children field shall be set to the number of devices along its branch, including itself. The number of requested addresses field shall be set to the sum of the numbers of requested addresses received from all children, plus one or some value larger than one if the device also wants to reserve some addresses.

Any device can update the number of children and/or the number of requested addresses by transmitting another children number report frame to its parent if it has not been assigned an address block. For example, if a leaf device becomes a non-leaf device, it should immediately send another children number report frame to update its parent. When a non-leaf device receives a children number report frame and if it has already reported the number of children, it shall immediately update its parent. If the non-leaf device has not reported the number of children to its parent, the received children number report frame shall be handled normally. If a device receives a children number report frame and it has already been assigned an address block, it shall not update its parent by transmitting another children number report frame. Instead, it should adjust the address assignment as described later.

After the root receives the information from all the branches, it should begin to assign addresses. During the address assigning stage, a top-down procedure is used. First, the root checks if the total number of devices in the network is less than the total number of addresses available. If not, address assignment fails. Next, the root assigns a block of consecutive addresses to each branch below it, taking into account the number of children and the number of requested addresses. The address block assigned to each branch is specified by the beginning address field and the ending address field given in the address assignment frame sent to each branch as described in 5.3.2.2.2. The actual number of addresses assigned could be less than or more than the number of requested addresses, but shall not be less than the number of children, depending on the availability of addresses. This procedure continues until the bottom of the tree. After address assigning, a logical tree is formed and each device has populated a neighbor list for tracking branches

below it. For example, Device C in Figure 38 can have a neighbor list as illustrated in Table 45 (not all fields shown). Note the begAddr of each entry represents the one-hop child device's address. The meanings of field Link Quality and Reliable Broadcast will be explained in the corresponding subclauses hereafter.

Table 45—Device C's neighbor list after address assignment

Beginning Address	Ending Address	Tree Level	Link Quality	Relationship	Reliable Broadcast	Number of Hops	…
6 (Device D)	8	3	188	child	TRUE	1	…
9 (Device E)	13	3	148	child	TRUE	1	…
14 (Device G)	14	3	111	child	TRUE	1	…
…							

The above neighbor indicates that Device C has total three branches; Device D owns address block [6, 7, 8] with its own address equal to 6; Device E owns address block [9, 10, 11, 12, 13] with its own address equal to 9; and Device G only owns one address, 14, which is its own address.

5.5.3.3 Adjustment of address assignment

More devices (therefore, more branches) are still allowed to be added at any level of the tree after address assignment, if additional addresses (reserved during address assignment) are available. Address assignment can be locally adjusted within a branch if a device runs out of addresses. For instance, a device can request more addresses from its parent. If the parent does not have enough addresses, it may either request additional addresses from its parent or adjust address assignment among its children. If there is a substantial change of the device number or network topology, which can not be handled locally, the network is allowed to go through the address assigning procedure again. In practice, address assignment is controlled by the application profile. For example, the application profile used for light control can specify that more addresses should be reserved for some special devices such as those near hallways. This improves the utilization of addresses and reduces the probability of address re-assigning as the network evolves. The intelligence of distributing limited addresses among all network devices is considered beyond the scope of this document.

5.5.4 Mesh topology discovery and formation

After a device has been assigned an address block, it shall broadcast several hello command frames to its neighbors, with the time to live (TTL) field of each hello frame set to *meshTTLOfHello* as defined in Table 42. By exchanging hello command frames, each device will build a link state table (LST) for all its neighbors within *meshTTLOfHello* hops as defined in Table 42. Each neighbor's address block is logged in the LST so that the whole branch below the neighbor is routable. Figure 39 shows a two-hop link state view of Device J. Note that Devices D, E, F, M, and N are not within 2 hops of Device J, but they are still directly routable since they are the children of those devices within 2 hops of Device J.

5.5.4.1 Link state generation

The link state table of a device, which consists of a *meshTTLOfHello*-hop neighbor list and a connectivity matrix, is updated upon the reception of each hello command frame.

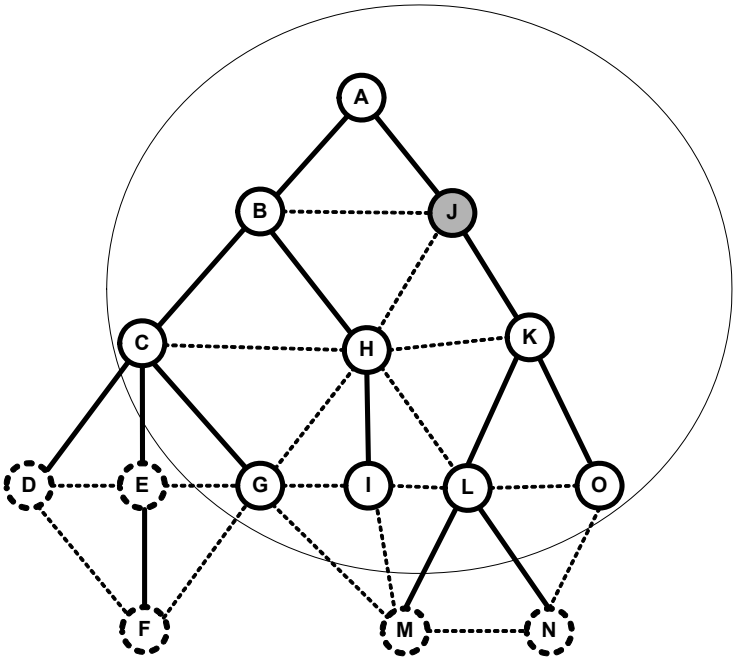

Figure 39—An example – two-Hop link state (view from Device J)

5.5.4.1.1 Neighbor list

Each device shall update its neighbor list, as illustrated in Table 46, upon the reception of each hello command frame. Not only the source of the hello command frame shall be added into the neighbor list, but also the one-hop neighbors of the source shall be added into the neighbor list unless the TTL in the incoming hello command frame is one. For those entries corresponding to the one-hop neighbors of the source, the Ending Address and Tree Level fields can not be populated from the incoming hello command frame and shall be marked as "unknown" temporarily. The "unknown" Ending Address and Tree Level fields will be replaced with actual values when a hello command frame is received from the corresponding neighbor. If no hello command frame is received from some neighbors during the entire hello command frame exchange procedure, a device can solicit for Ending Address and Tree Level information by broadcasting a neighbor information request frame, as described in 5.3.2.2.4, to its one-hop neighbors, including all the neighbors whose Ending Address and Tree Level are missing. Each one-hop neighbor received the message shall reply by transmitting back a neighbor information reply frame as described in 5.3.2.2.5 if it can provide the Ending Address and Tree Level information of one or more neighbors included in the neighbor information request frame.

Table 46—Neighbor list

Name	Type	Valid range	Description
Beginning Address	Integer	0x0000–0xfffe	The beginning address of the address block assigned to this neighbor. It is also the address of this neighbor.
Ending Address	Integer	0x0000–0xfffe	The ending address of the address block assigned to this neighbor.
Tree Level	Integer	0x00–0xff	The tree level of this neighbor
Link Quality	Integer	0x00–0xff	The link quality from the neighbor to this device. It is measured when the data frames are received from the neighbor and is ranged from from 0 (the lowest) to 255 (the highest).
Relationship	Enumeration	PARENT, CHILD, SIBLING DEVICE, SYNC-PARENT, SYNC-CHILD, NO_RELATIONSHIP	The relationship between this device and the neighbor.
Reliable Broadcast	Boolean	TRUE, FALSE	This field indicates whether the neighbor device supports reliable broadcast.
Status	Enumeration	UNKNOWN, DOWN, LEFT	The status of this neighbor.
Number of Hops	Integer	0x01–0xff	The hop distance between this device and the neighbor. It is calculated using the connectivity matrix described in the next subclause.
Group Membership	List of Integers	variable	A list of multicast groups of which the neighbor participates in.

5.5.4.1.2 Connectivity matrix

From the one-hop neighbor information included in each incoming hello command frame (except when *meshTTLOfHello* equals one), a device can construct a connectivity matrix for neighbors recorded in the neighbor list.

Table 47 illustrates one example.

The purpose of constructing a connectivity matrix is to calculate the value of the number of hops for each entry in a neighbor list. First the field number of hops of each device is set to infinity. Then, all devices directly connected to the current device (marked as “me” in Table 47) are one hop neighbors (nb_2, nb_{n-1}, …

in the example). Next, all devices directly connected to one-hop neighbors (and having a hops of infinity) are two-hop neighbors (nb_1, nb_3, … in above example). This procedure continues until hop numbers of all neighbors are populated. These hop numbers are then filled into the number of hops field for each entry in a neighbor list for data forwarding.

Table 47—An example of connectivity matrix

	me	**nb_1**	**nb_2**	**nb_3**	**…**	**nb_{n-2}**	**nb_{n-1}**	**nb_n**
me	–	–	+	–	…	–	+	–
nb_1		–	+	–	…	+	–	–
nb_2			–	+	…	–	–	–
nb_3				–	…	+	–	–
…					…	…	…	…
nb_{n-2}						–	–	+
nb_{n-1}							–	–
nb_n								–

NOTE 1— The plus or minus sign ("+" or "–") at the cross cell of two devices indicates they are or are not directly connected (i.e., they are or are not one-hop neighbors).

NOTE 2— For b-directional links, the matrix is symmetric, so only half of the matrix is needed as shown here.

NOTE 3—Hop information can be calculated using the connectivity matrix. Here we have:

1-hop neighbors: nb_2, nb_{n-1}, …

2-hop neighbors: nb_1, nb_3, …

3-hop neighbors: nb_{n-2}, …

4-hop neighbors: nb_n, …

5.5.5 Mesh path selection and data forwarding

5.5.5.1 Mesh path selection

The neighbor list is the only information base a device shall consult with when forwarding data frames. If the destination is one of the one-hop neighbors of the device, the data frame shall be sent directly to the neighbor. In other cases, a next hop device toward the destination needs to be found using the algorithm illustrated in the flow charts in Figure 40 and Figure 41.

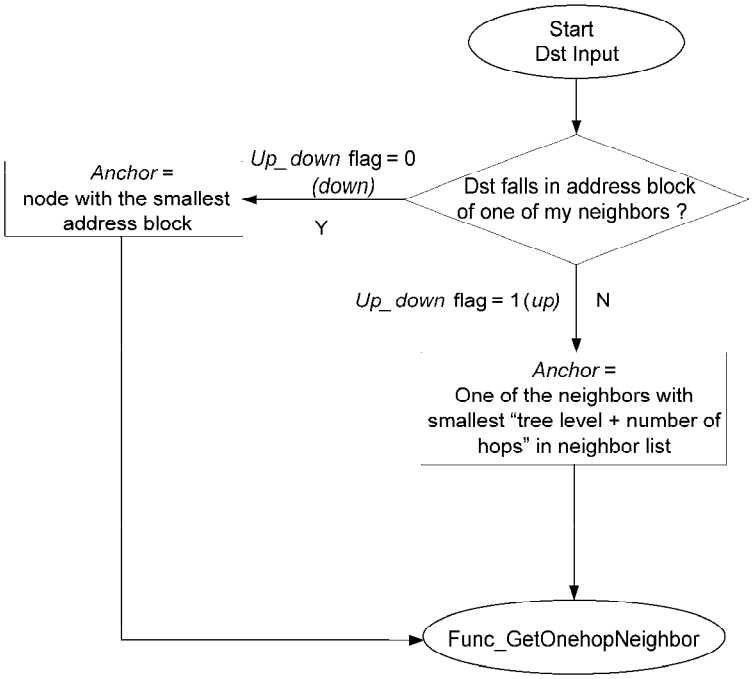

Figure 40—Flow chart of func_nextHop(*dst*)

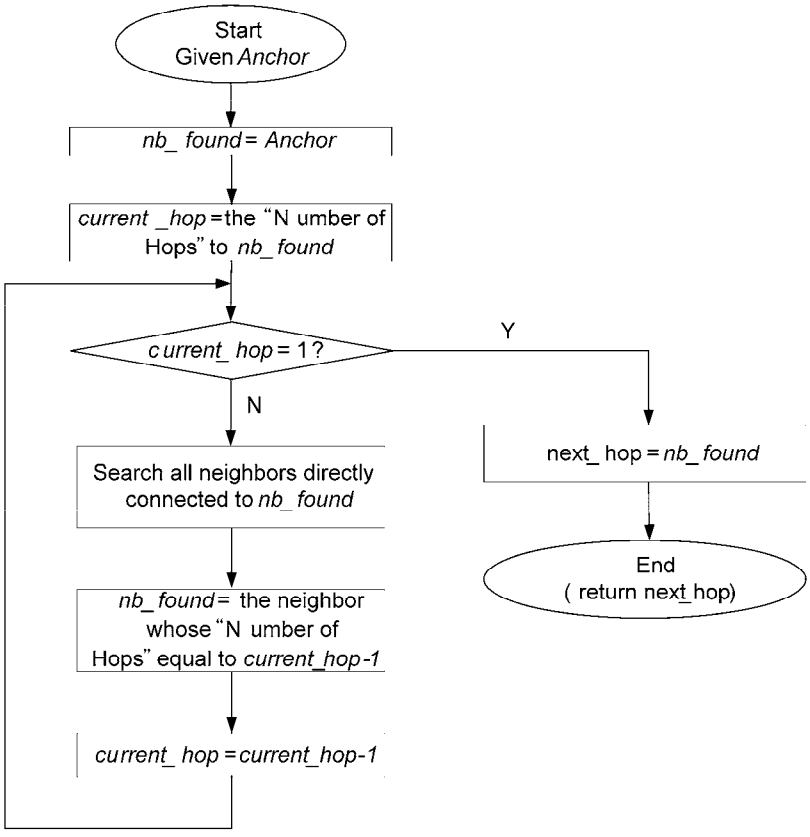

Figure 41— Flow chart of getOneHopNeighbor(*neighbor_found*)

When multiple neighbors are available for selection, as described in Figure 40 and Figure 41, a device may randomly select one neighbor for load balancing purposes. However, to mitigate "out of order" problems, a device should stick to one neighbor for a while once the neighbor is selected (rather than randomly select one neighbor each time).

5.5.5.2 Data forwarding

5.5.5.2.1 Reliability

Data forwarding supported by this recommended practice is not end-to-end reliable, i.e. acknowledgement from destination to source at mesh sublayer is not supported. When acknowledgement is requested by setting the AckTransmission field as described in 5.2.1.1, MAC layer reliable mechanism shall be used to achieve hop-by-hop reliability. To ensure the request of acknowledgement to be executed by each of the next hops, the request is carried by the mesh sublayer frame all the way to the destination in the mesh sublayer header as described in 5.3.1.1.4, the Transmission Options subfield of the Frame Control field.

5.5.5.2.2 Mesh sublayer payload

Due to the fact that mesh sublayer adds its header to the MHSDU it receives from the SSCS, the maximum allowed payload provided by the MAC is reduced by the size of the mesh sublayer header. Due to the fact that the length of mesh sublayer header varies, the safe length of MHSDU should be equal to or smaller than (*aMaxMACSafePayloadSize–meshcMaxMeshHeaderLength*), where the *meshcMaxMeshHeaderLength* is a mesh sublayer constant as defined in Table 41, which is equal to the maximum allowed length of the mesh sublayer header.

5.5.6 Mesh path maintenance

In this subclause, the mechanism of handling link breakage and recovery is described.

5.5.6.1 Sanity and consistency checking

After the link state generation stage, periodic hello command frames are broadcast for link state update. The sanity and consistency checking in this subclause does not need to be performed when hello command frames are periodically broadcast. However, for some network scenarios, hello command frames are not periodically broadcast, for instance, to reduce communication overhead and interference. Instead, hello command frames are only broadcast upon the detection of link failures, link recoveries, new neighbors, or for other purposes. If a device misses some hello command frames, its link state may not be accurate. Inaccurate link state may result in not only the selection of detoured routes but also, more seriously, routing loops.

The purpose for sanity and consistency checking is to help detect routing loops and link state mismatches. It works as follows. A device shall include in each data frame a one-bit Up-down Flag as described in the flow chart in Figure 40. After receiving a data frame, a device shall calculate $flag_1$ and $flag_2$ as follows.

$flag_1$ = the Up-down Flag value calculated by the transmitter using *meshTTLOfHello*-hop link state information and included in the received data frame
$flag_2$ = the Up-down Flag value calculated by the receiver of the data frame, using only (*meshTTLOfHello*-1)-hop link state information

The device shall check to see if $flag_1$ equals to $flag_2$. If $flag_1$ is not equal to $flag_2$, the receiver shall send a link state frame as described in 5.3.2.2.6, which contains its (*meshTTLOfHello*–1)-hop connectivity matrix, to the previous hop. Upon the reception of the link state frame, the previous hop shall compare the received connectivity matrix with its own and then broadcast a link state mismatch frame as described in 5.3.2.2.7 with a TTL set to the maximum hops of all neighbors in the frame. Any device having received a link state mismatch frame shall check if its address is included in the Addresses of Neighbors field and, if yes, shall broadcast several hello command frames to update the link state of its neighbors.

5.5.6.2 Link state maintenance

A device shall broadcast several hello command frames with increasing sequence numbers and with the TTL value set to *meshTTLOfHello* as defined in Table 42, if it detects its one-hop connectivity has changed due to link failures, link recoveries, or the detection of new neighbors.

Transmission failures can be caused by link failures (including device failures), collisions, or background interference. A neighbor to which a transmission has failed shall be first put into a probe list. A neighbor in the probe list has a state of either unknown or down. A neighbor with an unknown state is probed each *meshProbeInterval* seconds as defined in Table 42 after the last probe using a timer (timer-driven) or probed immediately each time it is selected to be the next hop of a data transmission (data-driven). Although the neighbor with an unknown state can still be selected as the next hop like a normal neighbor, it is not actually used for transmitting any data packet. All data packets having this neighbor as the next hop are buffered or dropped if without sufficient memory. The probe shall continue until the link to the neighbor is recovered or the total probe number, including both timer-driven probes and data-driven probes, reaches *meshMaxProbeNum* as defined in Table 42.

If a link is recovered, the corresponding neighbor shall be removed from the probe list and all packets buffered for this neighbor, if any, shall be forwarded to this neighbor. A link is considered recovered if a MAC layer ACK of a probe is received.

If the probe number reaches *meshMaxProbeNum* as defined in Table 42 before the link is recovered, the state of the neighbor shall be changed to down. The connectivity matrix shall be updated accordingly and hello command frames shall be broadcast with a TTL of *meshTTLOfHello* as defined in Table 42. After the broadcast of the first hello command frame, all packets buffered for the neighbor, if any, shall be routed via other routes. Data packets shall not be routed via other routes before the original next hop is determined down and at least one hello command frame has been broadcast to all *meshTTLOfHell*-hop neighbors.

The neighbor remains in the probe list if the link to the neighbor has been determined down, but it shall be probed only by timer (it shall not be used as the next hop of any data packet) and the probe interval shall be increased after each probe, up to a maximum value *meshMaxProbeInterval* as defined in Table 42. For example, a neighbor with a state of down may be probed using intervals 2, 4, 6, …, *meshMaxProbeInterval*, …, *meshMaxProbeInterval* seconds. This guarantees that, if the link recovers, it will be detected within no more than *meshMaxProbeInterval* seconds.

5.5.7 Leaving a mesh network

A device may leave a mesh network upon receiving a MHME-LEAVE.request primitive from its next higher layer or upon receiving a leave command frame from its parent device. The first case is called active leaving while the second is called passive leaving.

5.5.7.1 Active leaving

In the case of active leaving, the next higher layer of a mesh device decides to remove the device from a mesh network. To achieve this, the next higher layer issues a MHME-LEAVE.request primitive as defined in 5.2.2.10 to its mesh sublayer with the DeviceAddress parameter set to NULL.

If the RemoveChildren parameter is equal to TRUE, the MHME should first attempt to remove the device's children as described in 5.5.7.2 before leaving the network.

If the RemoveChildren parameter is equal to FALSE, the MHME should remove itself from the network. It first checks the multicast status of the device, if it belongs to one or more multicast groups. It should clear all the multicast-related information and prepare the corresponding fields to be put into the hello command frame. It then broadcasts a hello command frame, with the leaving network bit of the hello control field set to one, to its neighbors indicating it is leaving the network.

The MHME should then clear all of its references to the current mesh network, including connectivity matrix, neighbor list, group communication table, address mapping table, mesh sublayer buffer, multicast and broadcast transaction tables. It should also set all MeshIB fields to their default values and issue an MLME-RESET.request primitive to the MAC sublayer to reset the MAC sublayer PIB.

Finally, the mesh sublayer should issue a MHME-LEAVE.confirm to its next higher layer to indicate the completion of the leaving process.

When the hello command frame with the leaving network bit set to one reaches neighbors of the leaving device, the neighbors should first check their relationship with the leaving device.

If the hello command frame is received from a child device, the parent device should update its connectivity matrix and group communication table, if necessary. It should also change the status of this child device in its neighbor list to "left" and start a rejoin timer, *meshRejoinTimer*. The entry of the child device is kept in the neighbor list in case it rejoins the network after leaving. If the child device rejoins the network before the timer expires, it will be assigned the same mesh sublayer short address. When the timer expires and the child device has not rejoined the network, its entry should be deleted from the neighbor list and all references to this device should be removed. The parent device then sends its own hello command frame to update its neighbors with the change of link state and multicast group information.

If the hello command frame is received from a sibling device (neither a child device nor a parent device), the device should follow all the processes the parent device does except that it deletes the leaving device from its neighbor list immediately and remove all the references to the leaving device. It does not need to start a timer and wait for the leaving device to rejoin the network.

5.5.7.2 Passive leaving

In the case of passive leaving, a mesh device receives leave command frame to request it to leave the mesh network. To achieve this, the next higher layer of a device's parent issues a MHME-LEAVE.request primitive to its mesh sublayer with the DeviceAddress parameter set to the child device's mesh address.

The MHME of the parent device first determine whether the specified device is a child device. If the requested device does not exist, the MHME issues the MHME-LEAVE.confirm primitive with a status of UNKNOWN_CHILD_DEVICE. If the child device does exist, the parent device should attempt to remove it from the network by issuing a leave command frame to the child device. If the RemoveChildren parameter of the leave command is set to one then the device will be requested to remove its children as well. The mesh sublayer of the parent device should issue a MHME-LEAVE.confirm to its next higher layer indicating the child device has been requested to leave. However, the real leaving process will be conducted when the hello command frame from the child device is received.

Upon receiving the leave command, the child device should remove itself from the network following the same process of active leaving as described in 5.5.7.1. Basically it should send a hello command frame to all of its one-hop neighbors to indicate its leaving, clear all references to the mesh network at both the mesh sublayer and the MAC layer.

Finally, the mesh sublayer of the child device should issue a MHME-LEAVE.indication to its next higher layer to indicate the completion of the leaving process.

5.5.8 Mesh path selection and forwarding for multicast

The multicast routing protocol described in this subclause utilizes the logical tree and the local mesh links built by the unicast routing protocol as described in 5.5.5. The logical tree is a shared tree rooted at the mesh coordinator. When the tree is built, the neighbor information as well as its relationship (parent, child, or sibling) to a device is recorded in every device's neighbor list as defined in Table 46.

The goals of multicast routing can be described as building a logical tree that covers all multicast members of a group and updating the tree when devices join or leave the multicast group. In this scheme, joining and leaving the multicast group is allowed at any time during the multicast session. Thanks to the use of the tree structure, all control messages are unicast and no multicast routing table is needed. In most cases, the mesh coordinator is not bothered for transmitting control and data messages; hence, the congestion around MC and single-point-of-failure problem can be avoided or relieved. Furthermore, multicast data frames do not need to be sent to the mesh coordinator first. They can be propagated to all other members directly from the data sources to ensure simple and timely data delivery. Non-members can also send packets to members but not vice versa.

To better describe the multicast routing protocol, the entities are defined as follows:

- MC: the root of the routing tree of a mesh network. It keeps information of all multicast groups in the network. For each multicast group, the group address, the unicast address of the group coordinator, and the number of members are recorded.
- GM: a device participating in a multicast group. A GM should process any frames destinated to its group address and may send frames to its group. A device can be GM for multiple groups.
- Router (RT): a device on the multicast tree but not a GM. A router relays multicast frames for a multicast group.
- Multicast Agent (MA): a device with at least one of its end devices that is a member of a multicast group. A MA may or may not be a GM of the group itself.
- GC: a group coordinator is the controller of a multicast group. Each group should have its own GC and they can be different devices. A GC is also a GM.

They are illustrated in Figure 42.

For many reasons, there is a need for a central controller (the GC) for each multicast group. Its functions may include membership management, group key management/distribution and etc. These functions are considered out of scope of this document. For multicast routing, it is also helpful if the GC's address is known. In this case, join requests to a multicast group can be sent to the GC using unicast routing algorithm.

5.5.8.1 Group addressing

The group address is assigned from the next higher layer to the mesh sublayer through the MESH-DATA.request primitive as described in 5.2.1.1. The selection of the group address for a specific multicast group is considered beyond the scope of this document. By setting the transmission options field of the frame control field as described in 5.3.1.1 to multicast, the entire 16-bit short address space may be reused for multicast communication.

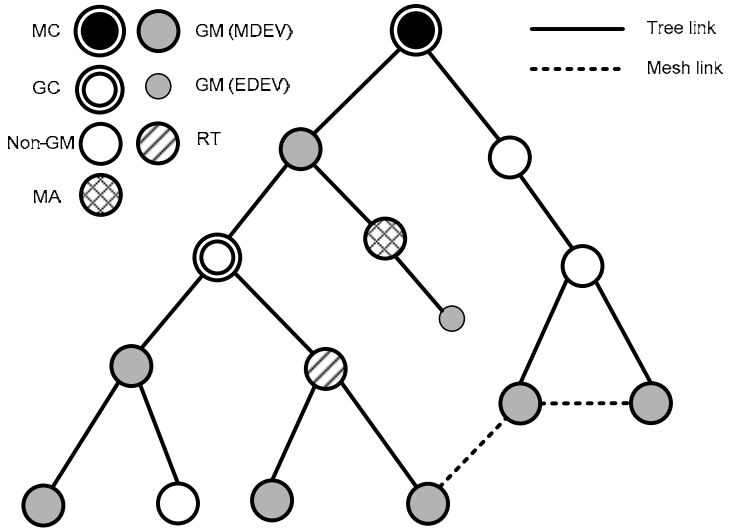

Figure 42—Entities of multicast protocol

5.5.8.2 Management of group membership

This subclause describes the functions related to membership management of multicast groups, which include joining and leaving the multicast group. To accomplish these functions, each device involved in multicast communications needs to maintain a group communication table as illustrated in Table 48. The group communication table is also refered to when routing the multicast data frames.

Table 48—Group communication table

Name	Type	Valid range	Description
Group Address	Integer	0x0000–0xfffe	The address of the multicast group this device is participating in.
Status	Bit field	As defined in Table 49	The role the device is playing in this multicast group. Dual stati may exist for a device, such as MC and GM, MC and GC.
Number of Links to the Group	Integer	0x00–0xff	The number of one-hop neighbors that are participating in this group.
List of Links to the Group	List of Integers	variable	The addresses of the one-hop neighbors that are participating in this group.

Table 49— Status field of group communication table

Bit	Name	Description
b_7	MC	This bit is set to one if the device is the MC; it is set to zero otherwise.
b_6	GC	This bit is set to one if the device is the GC of the multicast group indicated by the Group Address field; it is set to zero otherwise.
b_5	GM	This bit is set to one if the device is a GM of the multicast group indicated by the Group Address field; it is set to zero otherwise.
b_4	RT	This bit is set to one if the device is a RT of the multicast group indicated by the Group Address field; it is set to zero otherwise.
b_3	MA	This bit is set to one if the device is a MA and at least one of its child devices is a member of the multicast group indicated by the Group Address field; it is set to zero otherwise.
b_2–b_0	Reserved	–

5.5.8.2.1 Joining the multicast group

A device's process of joining a multicast group is initiated by receiving a MHME-MULTICAST-JOIN.request primitive from the next higher layer. The process could also be initiated when the device has a message to send to the multicast group and finds it does not have a route to that group. Both mesh devices and end devices can participate in group communications. Since end devices have limited resources and do not participate in routing, their joining process is different from that of the mesh devices and is described separately in 5.5.8.2.3.

Depending on their participating levels in a multicast group, there are two cases of joining; joining as the group coordinator (GC) or joining as a regular member.

If the device joining the multicast group is set to be the GC of this group (as indicated in the primitive), it should send a G-JREQ command with the destination address set to the MC's logical address. It should also have the MC as destination, Join as GC and G-JREP requested options set to one. The G-JREQ should be forwarded to the MC following the unicast algorithm (not necessarily following the tree links).

Intermediate devices which receive this G-JREQ should treat it as regular unicast command frame and forward it to the MC. When relaying the G-JREQ to the MC, intermediate devices do not need to cache backward routes in order to route the corresponding G-JREP back to the joining device.

Once the G-JREQ arrives at the MC, the MC checks its multicast group record for the specified multicast group. If the record for this multicast group does not exist, the MC creates an entry for this group in its multicast group record. Each entry of the multicast group record should consist of the multicast group address, the logical address of the GC and the number of members in the group.

If the record for this group does exist but with a different address of the GC, it indicates this joining device is to replace the current GC in the MC's record. The MC should update its record with this new GC information. If the record for this group does exist but with the address of GC empty and the number of group members larger than one, it indicates other members have tried to join the group before the MC.

The MC should then reply back a G-JREP toward the source of the G-JREQ with the joining device's unicast address as the destination address, the Join as GC flag set to one and the status field set to indicate the status of joining. Intermediate devices between the MC and the joining device should forward the G-JREP back to the joining device using the unicast routing algorithm. Intermediate devices don't need to change status to routers because the Act as router option is set to zero in the join options field.

When the G-JREP travels back to the joining device (i.e. the GC), the GC updates its Group Communication Table by creating a new entry for this multicast group and setting its status to GC. The joining device is now a member of the multicast group. It should send a hello command frame to its neighbors to update them its multicast group membership.

Except the GC, all other devices should join the multicast group following the process described hereafter.

The general joining process has the following steps:

(1) Search the neighbor list for existing members in the group.
    a. If existing member found, join through one of the members.
    b. If existing member not found, go to step (2).
(2) Check whether the GC's address is known.
    a. If GC's address is known, search the neighbor list for GC to
        i) If list, join through the GC;
        ii) If GC not found in the neighbor list, route the G-REQ to the GC using unicast.
    b. If GC is not known, go to step (3).
(3) Check whether the G-JREQ has reached the MC
    a. If this device is the MC, find the GC's address in its Multicast Group Records then go to (2)a
    b. If this device is not the MC, forward the G-JREQ to its parent device.

The devices don't need to record the backward routes under any circumstances because the group join replies can always find their routes back to the originator's of the G-JREQ using the unicast algorithm.

Whenever a member or the GC is found in the neighbor list, a device can send a G-JREP back to the source using the unicast algorithm and does not need to wait until the G-JREQ reaches the member or the GC. By using the unicast algorithm to send the G-JREP, instead of the route taken by the G-JREQ, mesh links can be utilized and the route could be shortened.

The following explains the joining process in details:

— STEP (1): Search the neighbor list for any member or router to join with

  If one or more group member or router is found, the device should choose one of the members that is the closest to it. When there are multiple member devices in the neighbor list, it is preferred to choose one of the tree links because tree links are established with the consideration of link quality and hence more robust.

  The device should then send a G-JREQ to the selected member using the unicast algorithm. The destination address of the G-JREQ should be set to the seleted member's address. Because the member is in the neighbor list of the joining device, when the joining device sends the G-JREQ, it can optionally set the G-JREP requested flag to zero, indicating it does not request a G-JREP from the member or any other devices in between them. The ACT as router flag of the G-JREQ should be set to

one if the G-JREP requested flag is to zero. In this case, the joining device should deem itself has joined the multicast group after it has sent the G-JREQ.

When the intermediate devices between the joining device and the existing member receive the G-JREQ, they first check the value of the Act as router flag in the G-JREQ. If the Act as router flag is set to one, intermediate devices should change their status to RT for this group as soon as they receive the G-JREQ. Otherwise, intermediate devices do not change their status until they receive the G-JREP. The intermediate device should then forward the G-JREQ to the selected member in the neighbor list of the joining device.

When the G-JREQ finally reaches the selected member, the member should reply with a G-JREP if the G-JREP requested flag is set to one. The selected member should also set the Act as router flag to one to indicate that intermediate devices receiving the G-JREP should change their status to router. The G-JREP will travel back to the joining device using the unicast routing algorithm. Intermediate devices should change their status to router after forwarding the G-JREP. The G-JREP will finally travel back to the joining device. If the G-JREP requested flag is set to zero, the selected member device should not send a G-JREP when it receives the G-JREQ.

After the joining device changes it status to GM and the intermediate devices change their status to RT, they should send a hello command frame to their neighbors to announce their capability of reaching the multicast group. For the routing purpose, there are no differences between members and routers. So even routers should announce they are members of a certain multicast group.

— STEP (2): Search the neighbor list for the GC if GC's address is known

The GC's address may be given in the MHME-MULTICAST-JOIN.request primitive or in the G-JREQ command frame received from other devices (when the GC as destination flag is set to one).

If the GC is found in the neighbor list, the joining device should send a G-JREQ to the GC using the unicast routing algorithm with the GC's address as the destination address in the command frame. The G-JREP requested flag may be set to zero and the Act as router flag set to one to join without waiting for the G-JREP.

If the GC is not found in the neighbor list, the device should continue to forward the to the GC device using the unicast routing algorithm. Even though the GC's address is known and is put in the destination address field, it doesn't mean that only the GC can reply with the G-JREP. Any member or router on the road may reply to the joining device in order for the device to quickly join the multicast group.

— Step (3): Check whether the G-JREQ has reached the MC if GC's address is not known

If the GC's address is not known and this device is not the MC, it should forward the G-JREQ to its parent device.

If the GC's address is not known but this device is the MC, it should search its multicast group record for the information of this multicast group. If group and GC information is not available, it indicates the GC has not presented in the network. The MC records the information (group address and GCs address) and increase the number of members counter, and sends a G-JREP with the Status field indicating the reason of join failure (GC not yet presented).

If the group and GC information is available, the device should replace the destination address with the GC's address in the G-JREQ and forward the G-JREQ to the GC.

As soon as a device has successfully joined a multicast group or an immediate device has become a RT for a multicast group, it should announce its reachability to this group. In this sense, a GM is not different from a RT because they are both connected to the multicast tree. A device announes the reachability by sending a hello command frame with the Hello control field set to indicate the newly joined multicast group.

Upon receiving the hello command frames with the multicast information included, devices should update their neighbor lists and the group communication tables.

The joining process can be summarized by the following flow chart as illustrated in Figure 43.

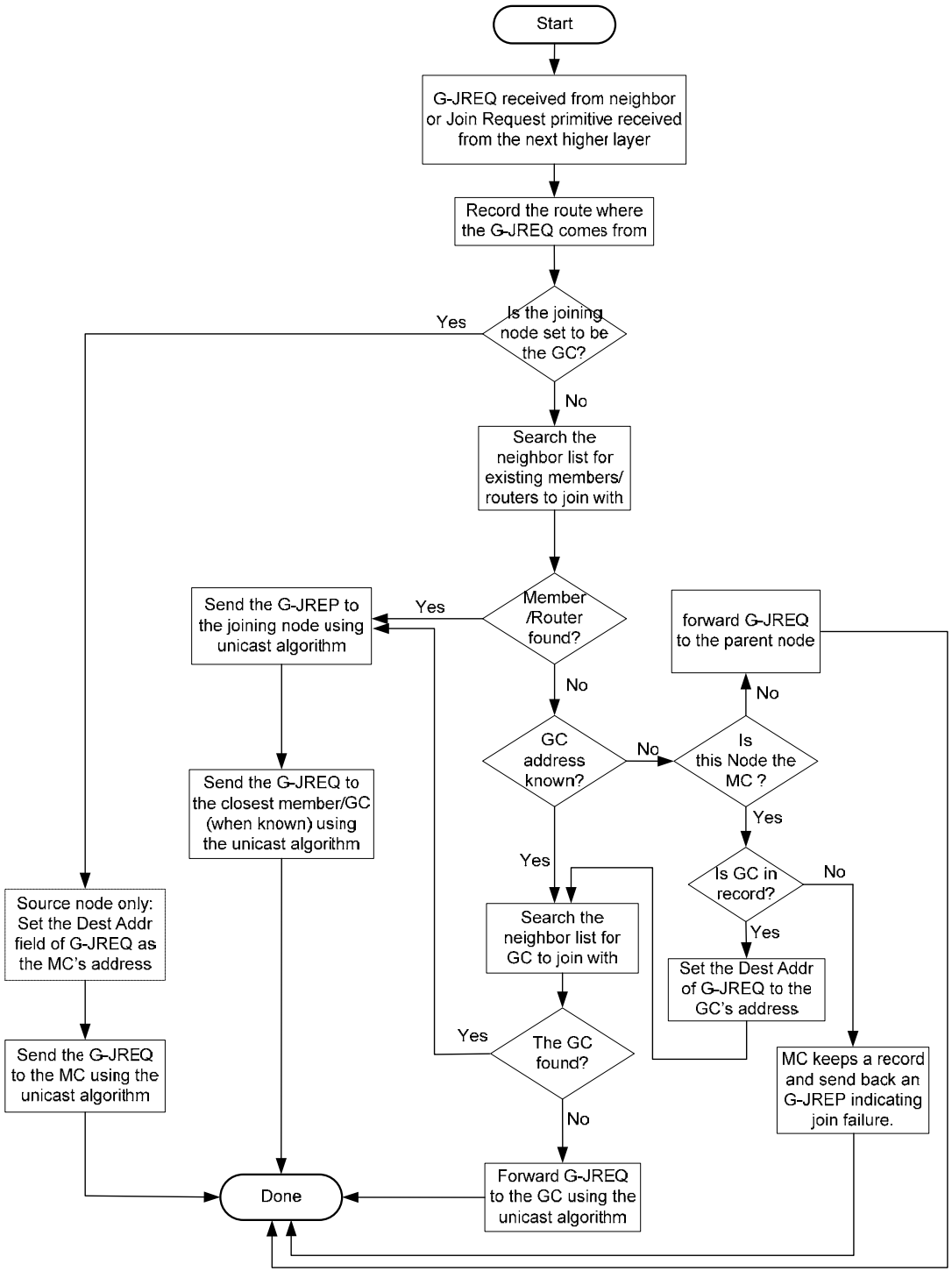

Figure 43— Flow chart of the joining process

Examples of different joining cases are illustrated below.

Figure 44 illustrates the case the GC joins the multicast group. Assume Device F is determined to be the GC of the group. It sends a G-JREQ all the way to the MC, Device A. The MC records the group and GC information, and then sends back a G-JREP all the way back to Device F. In this example it happens that the tree route F→E→C→B→A is the shortest path from F to A. In fact, Device F can take any shortest path between F and A, such as F→G→C→B→A, F→G→H→B→A, or F→G→H→J→A.

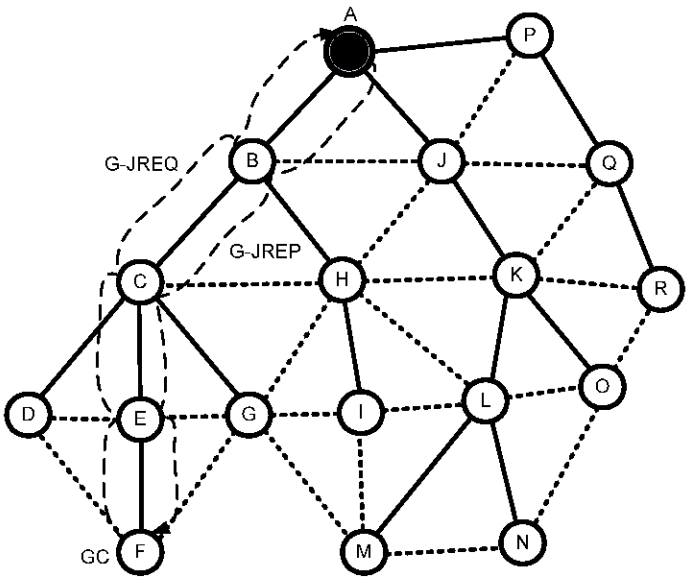

Figure 44— Join as the group coordinator

Figure 45 illustrates Step (1) of the joining process for regular nodes – members exist in the neighbor list Table 46. Assume the *meshTTLOfHello* is set to two. H is a member of a multicast group and Q is going to join this group without given the GC's address. Also assume H is the only member in Q's member list. Q first searches its neighbor list for the members of this group. Q finds H, which is two hops away from it. Q identifies two next hops, J and K, which have the shortest paths to H, using the unicast routing algorithm. Assume Q then selects J as its next hop to H and sends G-JREQ to J (with H's address as the destination address and the G-JREP requested flag set to one). J uses the similar approach as Q and finds H as its one-hop neighbor. Since Q can find H in its neighbor list, which indicates a route to H can be found without doubt, Q may optionally change its status to GM without requesting a G-JREP from H or intermediate devices. It does this by setting the G-JREP requested flag to zero and the Act as router flag to one. When receiving a G-JREQ with Act as router flag set to one, J knows it can now become a router of this multicast group even without receiving a G-JREP. Both Q and J should send a hello command frame locally to announce their involvement in the group.

Note although Q's three-hop neighbors, such as C, G, I, M, and N, are also in Q's neighbor list, Q doesn't know their multicast group membership. This is due to the fact the *meshTTLOfHello* is set to two therefore only devices within two hops can send their multicast group membership to Q.

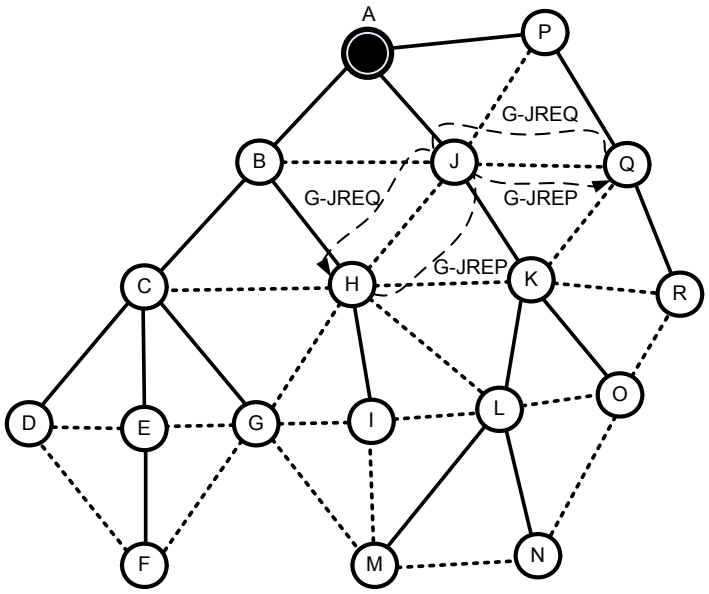

Figure 45— Joining example – members exist in the neighbor list

Figure 46 illustrates Step (2) of the joining process for regular nodes – GC's address is known. Assume O is the joining device and F is the GC, there are neither GMs nor RTs in the neighbor list of O, and F's unicast address is known to O when it attempts to join the Group. Since O cannot find any GMs and GC in the neighbor list, O forwards the G-JREQ using the unicast routing algorithm. O does this by sending the G-JREQ to its parent K. K searches it neighbor list for GMs in this group and fails to find one. K then continues to search the GC in its neighbor list. This time K is able to find the GC, Device F, is in its neighbor list. K may send a G-JREP back to O and forward the G-JREQ toward Device F with the G-JREP requested flag set to zero and the Act as router flag set to one. All devices between O and F (i.e. Device K, H, and G) become RTs and O becomes a GMs.

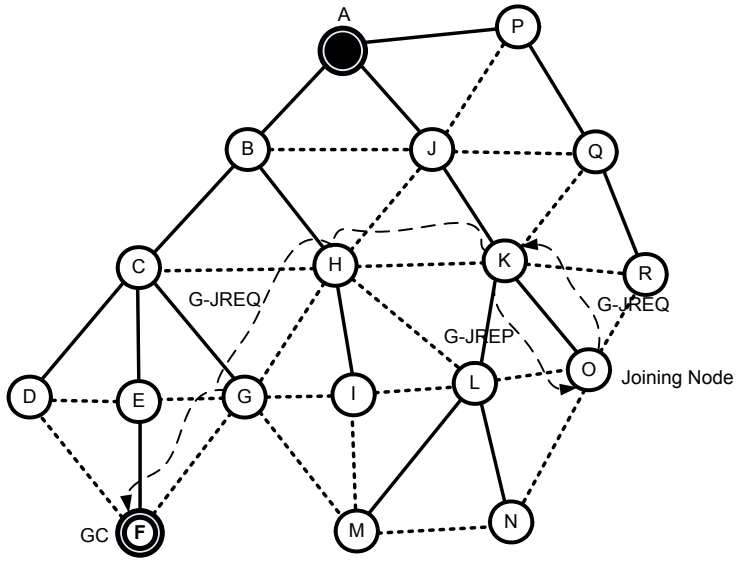

Figure 46— Joining example – the GC's address is known

Figure 47 illustrates Step (3) of the joining process for regular nodes when GC's address is unknown.

Assume Q is the joining device, F is the GC and the only GM in the network for this group. Also assume there are neither GMs nor RTs in the neighbor list of Q. Comparing to the previous example illustrated in Figure 46, the difference is the GC's address is not known to the joining device, Q. Therefore, the only thing Q can do is to send a G-JREQ to its parent, P, with the G-JREP requested flag set to one and the Act as router flag set to zero. P finds neither a GM nor the GC in its neighbor list, so it forwards the G-JREQ to its own parent device, which in this case happened to be the MC, Device A. Since the GC has already registered with the MC, the GC's address is known. So from now on the G-JREQ will have the GC's address in the destination address field and will be forwarded to the GC using unicast algorithm.

When the G-JREQ reaches B, B finds the GC (Device F) is in its neighbor list. B can decide whether it should wait for the G-JREP from the GC or reply with its own G-JREP back to the joining device. Assume B decides to reply with its own G-JREP back to the joining device. It should send the G-JREP using the unicast algorithm with the Act as router flag set to one. In this example, the G-JREP takes the route B→J→Q using its neighbor list instead of following the path the G-JREQ took (B→A→P→Q). This is the reason the backward routes to the joining device are not recorded – a better route may be found from the other direction.

At the mean time, B should continue to forward the G-JREQ to the GC using the unicast routing algorithm. The G-JREP requested flag may be set to zero and the Act as router flag may be set to one so that C and E will change their status upon receiving the G-JREQ.

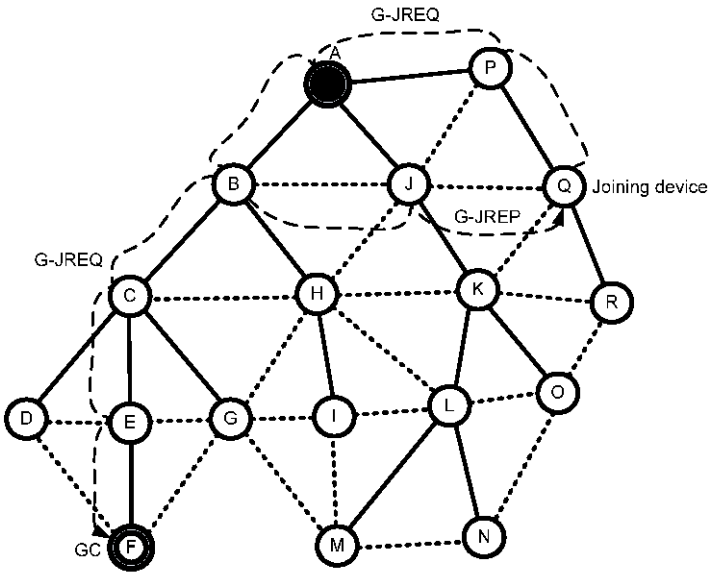

Figure 47 — Joining example – the GC's address is unknown

5.5.8.2.2 Leaving the multicast group

This subclause defines the leaving procedure of mesh devices. The leaving procedure of end devices is defined in 5.5.8.2.3.

A GM can leave a group at any time. However, before it leaves a group, it first has to check whether it is a leaf device of the multicast tree. To do this, it checks the number of members among its one-hop neighbors which belong to this group as indicated in the Number of links to the group field of the group

communication table as defined in Table 48. If the number is one, it can leave the group. Otherwise, it can only change its status to RT and still remain in the multicast tree. In this case, no hello command frame needs to be sent. The device only needs to change its status in the group communication table.

If a device is a leaf device and decides to leave, it should broadcasts a hello command frame to its neighbors with the Hello control field set to indicate Newly left multicast group and the multicast group address included in the multicast address list field. Neighbors receiving this hello command frame should update their neighbor table to reflect the fact that they are no longer able to reach the multicast group through this device.

Whenever a one-hop neighbor leaves a multicast group or becomes unreachable, a device should reduce the Number of links to the group field in the group communication table by one. When this number becomes zero, the device has lost the connection to the multicast group. It should restart the process of joining the group and try to reconnect to the multicast tree.

5.5.8.2.3 Joining and leaving procedure of end devices

End devices have limited resources so they should join and leave multicast groups through their parent devices. In this case, the parent devices should act as their MAs. End devices should only be GMs and never be the GC and RTs.

When joining a multicast group, an end device simply sends a G-JREQ with the Group Address field set to the multicast group address it wishes to join to its parent device. Upon receiving the G-JREQ, if the parent device is already the GC, a GM, or a RT of the same multicast group, it should reply with a G-JREP. Upon receving the G-JREP, the end device has successfully joined the multicast group.

If the parent device is not related to the multicast group, it should initiate its own joining procedure on behalf of its end device. It should follow the procedure defined in 5.5.8.2.1 for joining except that it should not become a GM of the multicast group. Once it has successfully joined the multicast group, it should set both the RT and the MA bits in the Status field in its group communication table to one.

When leaving a multicast group, an end device sends a G-LREQ with the Group Address field set to the multicast group address it wishes to leave to its parent device. Upon receiving the G-LREQ, the parent device reply with a G-LREP back to the end device. If the parent device itself has no relation with the multicast group (not a GM, a RT, or the GC) and the end device is the only GM in this multicast group, the parent device should also leave the multicast group following the procedure defined in 5.5.8.2.2.

Upon receiving the G-LREP, the end device should clear all the information related to this multicast group and it has now left the group.

5.5.8.3 Data transmission

The data transmission mechanisms are different for mesh devices and end devices.

5.5.8.3.1 Data transmission of mesh devices

Upon receiving the request of transmitting a multicast data frame from the next higher layer, the source device's mesh sublayer puts the multicast group address as the destination address in the mesh sublayer header and indicates to its MAC layer that the destination address of the MAC frame should be the broadcast address. The source device's MAC layer broadcasts the data frame by setting both the destination PAN ID and MAC short address to broadcast address (0xffff in the case of IEEE Std 802.15.4-2006).

All neighbors receiving this broadcast data frame (at MAC layer) will pass it to their mesh sublayers. Unlike traditional routing protocol, a routing table is not needed for multicast data forwarding. The decision of forwarding or not depends on the status (e.g. GM, RT, and etc.) of the device in the multicast group. When a multicast data frame is received from the next higher layer or from a neighbor device, the device first checks its status by looking up the group communication table.

If the corresponding entry exists and the device is the GC, a GM, a MA, or a RT of this multicast group, it should relay the frame by broadcast at the MAC layer.

If the corresponding entry does not exist or if the entry exists but the device is neiter the GC nor a GM nor a MA or nor a RT, the device should check where the data frame comes from. If the frame is from a neighbor device, it should discard the frame because it is not on the multicast tree and should not forward the data frame for this multicast group. If the frame is from its next higher layer, it indicates the device wants to be a member of this group. The device should initiate the joining process to join the multicast group first and then relay the frame. The data frame should be stored in the buffer of this device and forwarded after the device joins the multicast group.

To reduce the broadcast traffic at the MAC layer, a device should check the number of one-hop neighbors which are GMs or MAs or RTs of the multicast group before forwarding the data frame (using the multicast communication table).

— If the number is one, then this device is a leaf device and the data frame must be received from its own next higher layer. It should just unicast the data frame to its next hop member.

— If the number is two (including the neighbor from which it received the frame), it has to determine whether the data frame is from its own next higher layer or received from other devices. If the frame is from its next higher layer, it should broadcast the data frame to its next hops; otherwise, this frame is from one of its two member neighbors. It may choose to unicast the data frame only to the other next hop member.

— If the number is more than two, the device should just broadcast the data frame at the MAC layer.

NOTE—When some mesh devices in the network are running in the low-power mode, broadcast takes more effort (longer delay) to be achieved. Therefore it is better to use unicast whenever it is possible.

To prevent duplicate data frames when propagating the multicast frames, a multicast transaction table (MTT) is maintained in each multicast-related device. Each entry of the MTT is a multicast transaction record (MTR), which records the following three items of the last seen multicast frame. A newly received data frame should be silently discarded if its values of these three items are the same as recorded in the MTT.

Table 50 — Multicast transaction record

Name	Type	Valid range	Description
Group address	Integer	0x0000–0xfffe	The address of the multicast group this device is participating in.
Source address	Integer	0x0000–0xfffe	The 16-bit logical address of the device that injects the data frame into the network.
Sequence number	Integer	0x00–0xff	The sequence number of the multicast data frame.

5.5.8.3.2 Data transmission of end devices

If a GM is an end device, it should only generate and receive multicast data frames and should not forward the frames for other GMs. When generating data frames, they are sent directly to the end device's MA, which is its parent device. The Destination Address of the mesh sublayer data frame should be set to the multicast group address and the Destination Address of the MAC layer data frame should be set to the parent device's logical address. When the parent device receives the data frame, it should forward the data frame following the procedure defined in 5.5.8.3.1.

When the mesh sublayer of an end device GM receives a multicast data frame, it should pass it to its next higher layer without forward it further.

5.5.9 Reliable broadcast

This clause describes the fundamental procedures for reliable transmission and reception of broadcast frames. The WPAN mesh sublayer provides reliable broadcast service to the next higher layer. Reliable broadcasting service may be used for applications such as software update and any control commands that require feedback. This service is optional for this version of the recommended practice. The implementation of the functions that support reliable broadcast is up to the discretion of the implementers.

5.5.9.1 Transmission, reception, and acknowledgment of the broadcast data frame

When transmitting a data frame via reliable broadcasting, the next higher layer should set the destination address mode in the MESH-DATA.request primitive to indicate the short address is in use and the destination address set up to the logical broadcast address, *meshcBroadcastAddress.* In addition, the device should set both BcstTransmission and ReliableBcst parameters to TRUE to indicate a reliable broadcast.

At the mesh sublayer, a device that intends to broadcast a data frame reliably should set the protocol version to one, the frame type to zero, the destination address to the logical broadcast address, *meshcBroadcastAddress*, and the source address to the logical address of the source, in the mesh sublayer header. The device should also set transmission options vector to 0x01 and 0x40 to indicate a reliable broadcast. The sequence number for the frame should be generated randomly and incremented by one afterwards. Immediately after transmitting the broadcast data frame, the device should start the *meshRBCastTxTimer* described in Table 42.

A device that broadcasts or relays broadcast frames should maintain a broadcast transaction table. This table can be shared by both reliable and non-reliable broadcast transactions. For unreliable broadcast, only the originator's address and the sequence number need to be recorded to avoid duplicate processing of the broadcast data frames. For reliable broadcast, the acknowledgement (either by overhearing the re-broadcast of the same data frame or by receiving an acknowledgement frame for that data frame) from every neighbor device needs to be obtained for the transmission to be considered successful. Therefore, besides the originator's address and the sequence number, each device has to maintain a bitmap of one-hop neighbor's acknowledgement. The bitmap should have the size (number of bits) equals to the number of one-hop neighbors in the neighbor list and is initialized to zero. The bitmap is arranged in a way that the i^{th} bit in the bitmap corresponds to the i^{th} one-hop neighbor entry in the neighbor list (when the neighbor list also includes extended neighbors, make sure one-hop neighbors are listed first for easy indexing). When a data frame or a data frame as an ACK (with the same sequence number as the current broadcast data frame) is received, the device should set the corresponding bit to one. Only when all bitmap fields are set, should the device transmit or relay the next broadcast data frame. An example of the broadcast transaction table is illustrated in Table 51.

Table 51 — An example of the broadcast transaction table

Originator address	Sequence number	Reliability flag	Bitmap for reliable broadcast
srcAddr1	m	TRUE	$b_7b_6b_5b_4b_3b_2b_1b_0$
srcAddr2	n	FALSE	
…			

Upon receiving the broadcast data frame successfully (the sequence number matches what the receiving device is expecting), the receiving device first needs to record the data frame in the broadcast transaction table, if it is the first data frame received. The mesh sublayer payload will then be forwarded to the device's next higher layer for processing. If reliable delivery is required, the receiving device has to also set the corresponding bit in the bitmap to indicate the successful reception of this specific data frame from the sender. For unreliable broadcasts, duplicate data frames will be silently discarded; for reliable broadcasts, duplicate data frames will set their corresponding bits in the bitmap before being discarded.

The receiving device will then determine whether it should relay the broadcast frame. Devices that find at least one of the bitmap fields remaining at zero should forward the frame to their one-hop neighbors. The frame forwarded back to the originator or sender is implicitly interpreted as an acknowledgment (ACK), which does not entail use of an extra ACK frame. The time to forward the received frame is randomized using a random timer to reduce the possibility of collision (this mechanism may also apply to non-reliable broadcast). The timer is generated in the range of [0, *meshRBCastRxTimer*] where *meshRBCastRxTimer* is a MeshIB attribute described in Table 42. After forwarding the data frame, the device should set its transmit timer to *meshRBCastTxTimer* described in Table 42. Randomization of the transmission time of a frame reduces the possibility of collisions. After forwarding the frame, the device should set its transmit timer to *meshRBCastTxTimer* described in Table 42.

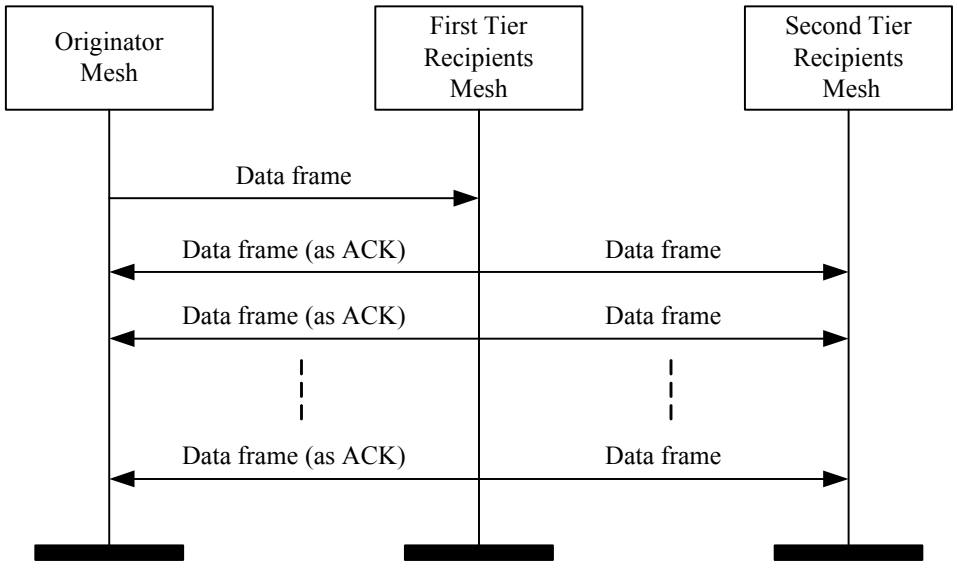

Figure 48 — Use of data frame for acknowledgement

5.5.9.2 Transmission and reception of the rebroadcast data frame

If any one of the bitmap fields remains zero when the *meshRBCastTxTimer* expires, the device should rebroadcast the data frame and reset the *meshRBCastTxTimer* to its initial value. If the device tries *meshMaxRBCastTrials* times for rebroadcasting the data frame and still finds zero in any of the bitmap fields, the device should give up rebroadcasting and update the status of the corresponding neighbor device(s) to unknown. The MHME should then start probing the neighbor(s) using the mechanism described in 5.5.6 .

5.5.9.3 Interoperability among reliable and non-reliable broadcast devices

Because the function of reliable broadcast is optional in this version of the recommended practice, there are chances a mesh network may consist of devices with or without the capability of providing reliable broadcast service. In this case, the following rules may be followed to ensure the interoperability among these devices.

— Non-reliable-broadcast-capable devices are allowed to join a network that is reliable-broadcast-capable. The newly joined device will degrade the reliable broadcast service to a level between reliable and non-reliable broadcast.

— All one-hop neighboring devices need to know and record the fact that the newly joined device is non-reliable-broadcast-capable.

— When a reliable broadcast frame is sent from a reliable-broadcast-capable device to a non-reliable-broadcast-capable device, neither explicit acknowledgement (by ACK frame) nor implicit acknowledgement (by overhearing the retransmission) is expected and no retransmission is required. In the implementation, the reliable-broadcast-capable device should not include the non-reliable-broadcast-capable device in its bitmap.

— When a reliable broadcast frame is sent from a non-reliable-broadcast-capable device to a reliable-broadcast-capable device, the latter will not do explicit acknowledgement (sending ACK frames).

5.5.10 Energy saving in battery-powered networks

In many applications, mesh devices are expected to have low data rates and to be operated on battery. IEEE Std 802.15.4-2006 provides power saving mechanisms with the superframe structure in the beacon mode. However, the mesh sublayer is designed on a non-beacon mode for the flexible mesh communication. Thus, the synchronized superframe structure of IEEE Std 802.15.4-2006 cannot be utilized. For a long battery life, efficient energy saving mechanisms should be supported. This subclause describes the mechanisms for power saving at the mesh sublayer. Two approaches are described in this subclause, one asynchronous approach and one synchronous approach. Both approaches operate at the non-beacon mode of IEEE Std 802.15.4-2006. An active duration and an inactive duration are specified by the command frames at the mesh sublayer and periodically repeated. Implementing the function that supports low power feature is at the implementer's discretion.

5.5.10.1 Asynchronous approach

Asynchronous energy saving (ASES) for mesh sublayer is a mechanism of power saving which utilizes an asynchronous time schedule for data transmission. Under ASES mode, the time structure of a device has an active and an inactive duration, which is similar to the superframe structure in the IEEE Std 802.15.4-2006 beacon mode. Similar to a beacon frame, a command frame named wakeup notification (WN), is transmitted at the beginning of the active duration of each superframe. However, the active durations of devices are not synchronized.

For the transmissions of data frames, ASES uses two different mechanisms. For unicasting, a device uses a receiver oriented mechanism. A device waits for the active duration of the receiver. Then, it transmits a frame within the active duration. When it comes to broadcasting, a device uses a sender oriented mechanism. It wakes all neighbors up by transmitting command frames for longer time duration than a wakeup interval. Then, it broadcasts the data frame.

5.5.10.1.1 Requirements

ASES utilizes some MAC layer primitives and PIB variables to control the MAC layer and requires a mesh sublayer timer with 1-ms resolution as the reference clock.

ASES uses the following set of MAC primitives: MCPS-DATA, MCPS-PURGE, and MLME-SET. To transmit or receive a command frame, MCPS-DATA is used. To clear out a pended frame MCPS-PURGE is used. To turn on/off the receiver circuitry, MLME-SET.request is used. By setting the *macRxOnWhenIdle* attribute in MAC PIB to FALSE, the mesh sublayer turns off the receiver circuitry. In addition to *macRxOnWhenIdle, macMinBE and macMaxCSMABackoff* need to be adjusted by the mesh sublayer for the ASES mode.

Although the standard primitive interfaces of IEEE Std 802.15.4-2006 provide substantial control means, the efficiency of ASES relies on the close cooperation from the MAC layer. For maximum energy saving, a prompt response from the MAC is essential.

In addition, ASES requires a mesh sublayer timer. The basic time unit is defined as *meshcTimeUnit*, which is equal to 1ms. All timing parameters defined in ASES use *meshcTimeUnit* as a time unit, and all operations are based on the mesh sublayer timer.

5.5.10.1.2 Time structure

A time line of a device is divided into wakeup intervals. A wakeup interval is subdivided into an active duration and an inactive duration. In the active duration, a device announces that it is in the active duration by transmitting a wakeup notification command. Then, it waits for possible frame transmission. In the inactive duration, a device enters a low power (sleep) mode by turning off the receiver circuitry.

The time structure is defined by the following variables: *meshWakeupOrder*, *meshActiveOrder,* and *meshBaseActiveDuration*. The MeshIB attribute *meshWakeupOrder* describes the interval of the periodic wakeup. The value of *meshWakeupOrder*, *WO*, and the wakeup interval, *WI*, are related as follows: for $0 \leq WO \leq 14$, $WI = meshBaseActiveDuration * 2^{WO}$ ms. If *WO* is equal to 15, a device is always in the active duration without transmitting WNs. Therefore the wakeup interval does not exist. The value of *meshActiveOrder* is ignored if *WO* is equal to 15.

The MeshIB attribute *meshActiveOrder* describes the length of the active duration within the wakeup interval. The value of *meshActiveOrder*, *AO*, and the active duration, *AD*, are related as follows: for $0 \leq AO \leq WO \leq 14$, $AD = meshBaseActiveDuration * 2^{AO}$ ms.

An example of a time line is illustrated in Figure 49.

All devices should have the same value of the wakeup interval, but they may have different values of active durations. The active durations of devices are not synchronized. To notify that it is in the active duration, each device transmits a wakeup notification (WN) command at the beginning of the active duration. In WN, the wakeup interval and the active duration are announced. Therefore, if a device turns its transceiver on for longer than one wakeup intervals, it is able to receive a WN from a neighbor. After receiving the WN, it knows not only that the neighbor device is in the active duration, but when the neighbor goes to sleep again.

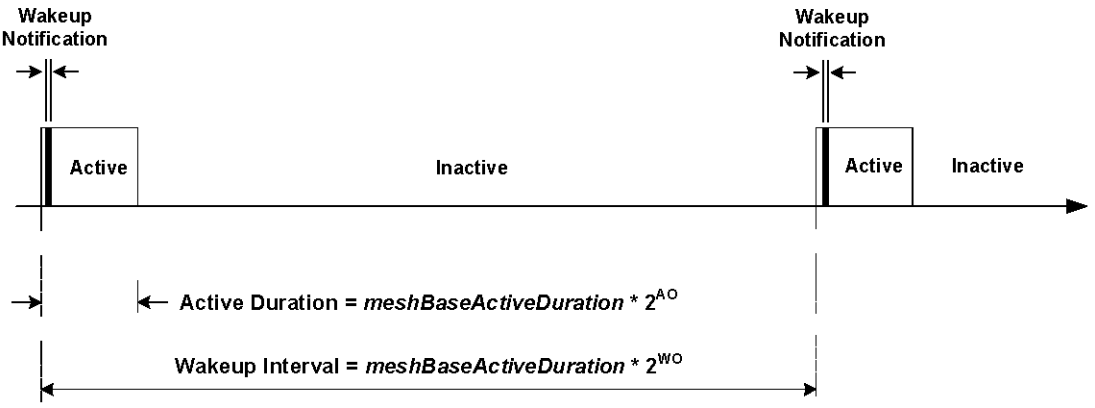

Figure 49— Time structure for ASES

5.5.10.1.3 Frame transmission

Frame transmission of ASES is controlled by two variables: *meshASESExpected* and *meshASESOn*. *meshASESExpected* controls the transmission method for start/join procedure. If the value is TRUE, a device assumes that the network is running on the ASES mode. Then, it follows as ASES initialization procedure defined in 5.5.10.1.4. If it is FALSE, the initialization precedes without considering ASES.

meshASESOn decides whether a device is in use of the ASES mode or not. After joining, a device transmits WNs periodically as defined in *meshWakeupOrder* if the device sets *meshASESOn* to TRUE. In addition, the device has an active and an inactive duration as defined in *meshActiveOrder*. Since active durations of devices may not be synchronized, all transmissions should follow one of the methods defined in this subclause. If the value of *meshASESOn* is FALSE, all frames are transmitted without considering ASES.

Unicast transmission is started when an MESH-DATA.request is issued. If *meshASESOn* is TRUE, the mesh sublayer turns on the receiver circuitry, and stays awake until it receives a WN from the destination device. Upon receiving the WN, the device copies the value of *meshActiveOrder* in the payload to *meshDestActiveOrder, DO*. After receiving the WN, the device estimates the destination is in an active duration if the elapsed time of the mesh timer is less than the time determined by *meshBaseActiveDuration* $* 2^{DO}$.

Within the active duration of the receiver, the transmitter transmits using one of the following two transmission methods. When the remaining active duration of the receiver is long enough to transmit a data frame, the transmitter transmits a data frame using MCPS-DATA.request. If the remaining active duration is not enough for a long data frame, but longer than or equal to 3**meshcTimeUnit*, the transmitter transmits a short command frame named extension request, EREQ, which is to extend the active duration of the receiver. An EREQ is a broadcast command frame at the MAC layer containing the destination address in the mesh sublayer payload. Broadcasting enables the mesh sublayer to detect activity on the channel without filtering at the MAC layer.

The device, upon receiving an EREQ, checks the destination address. If the address matches its own mesh address or 0xffff, the device extends the active duration by the time duration defined in *meshEREQTime*. Then, the device replies with an extend reply command, EREP, to the transmitter of EREQ if the EREQ had its own address. The transmitter that issued EREQ transmits a data frame after receiving the EREP.

As an example, a device may have the minimum active duration, and the duration may be shorter than a long data frame. To transmit a long data frame to the device, the transmitter should request an active time

extension. Therefore, when devices set AO to zero, handshaking with an EREQ and an EREP will be a typical method.

If a device receives an EREQ for unicast but with different address from own address, it may simply ignore the frame in most cases. However, if a device has a packet to transmit, but receives that EREQ during the active duration of the destination, the device defers the transmission to the next active duration of the destination. For the purpose, the mesh sub-layer may issue MCPS-PURGE.request to cancel the EREQ transmission. Then, the layer increases the ASES retry counter, ARC, by one. If the ARC is less than *meshMAXNumASESRetries,* the mesh layer retries by either one of following ways. One is waiting for the next WN while turning on the receiver circuitry. The other is estimating the next WN time, and turning off the receiver circuitry until the estimated time. If ARC is equal to or greater than *meshMAXNumASESRetries*, a mesh layer issues MESH-DATA.confirm with status, CHANNEL_ACCESS_FAILURE.

Examples of the ASES transmission are presented in Figure 50. The upper timelines show the transmission when the active time duration is long enough and the lower timelines show the usage of an EREQ and an EREP.

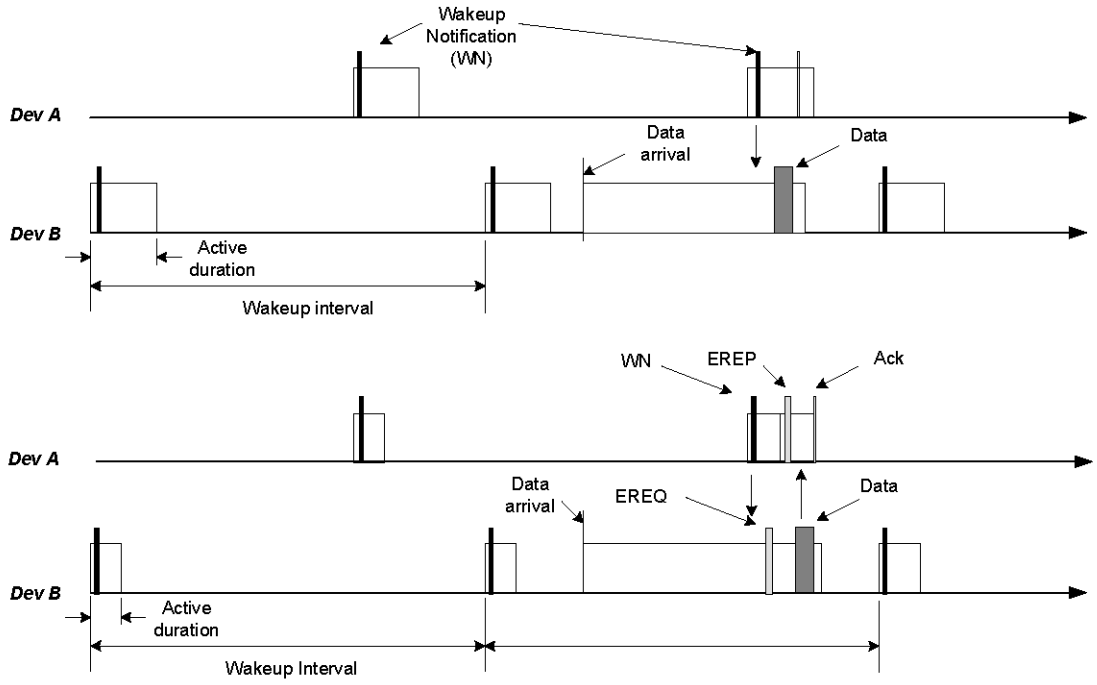

Figure 50—Examples of ASES unicast – Dev B transmits a frame to Dev A

Figure 51 illustrates a sequence of messages that may be used by a device in ASES.

Due to the nature of the error-prone wireless channel, the transmission procedure is not always completed successfully. The following are the possible scenarios that lead to unsuccessful transmissions.

— *Loss of WN*: When a mesh sublayer has a frame to transmit, it turns on the receiver circuitry to wait for a WN of the destination. At this time, the mesh device sets a timer that will expire after one wakeup interval. If the timer is expired before receiving a WN, the device increases the ASES

retry counter, ARC, by one. If the ARC is less than *meshMAXNumASESRetries,* the mesh sublayer retries by waiting for a WN again. If the ARC is equal to or greater than *meshMAXNumASESRetries*, the mesh sublayer issues MESH-DATA.confirm with status, TRANSACTION_EXPIRED. The ARC is cleared after issuing MESH-DATA.confirm.

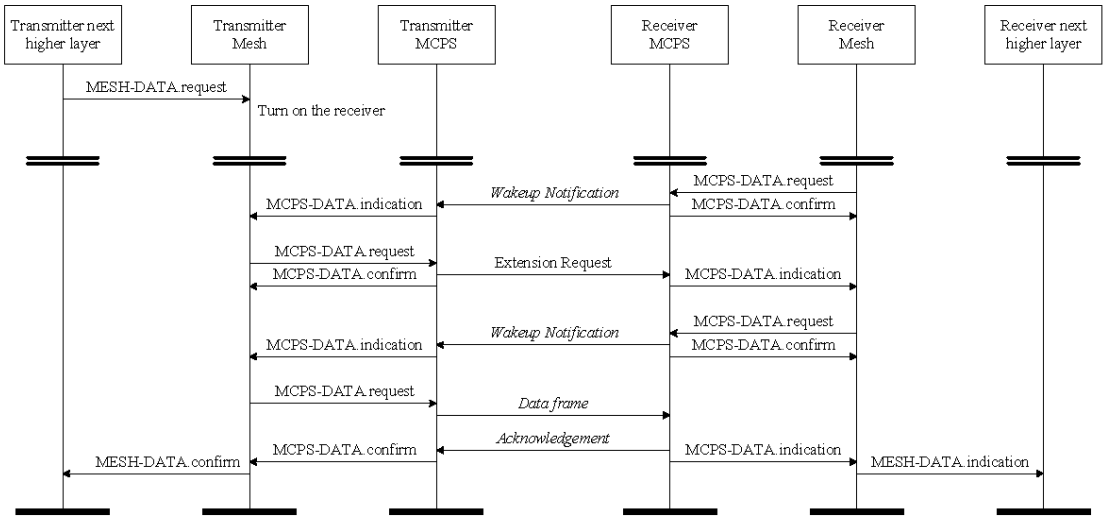

Figure 51—Message sequence chart for ASES unicast

— *Loss of EREP*. When a mesh sublayer transmits an EREQ, it sets a timer that will expire after *meshEREPTime*. If the timer is expired before receiving an EREP, the device increases the ARC by one. If the ARC is less than *meshMAXNumASESRetries,* the mesh sublayer retries by either one of following ways. One is waiting for the next WN while turning on the receiver circuitry. The other is estimating the next WN time, and turning off the receiver circuitry until the estimated time. If ARC is equal to or greater than *meshMAXNumASESRetries*, a mesh sublayer issues MESH-DATA.confirm with status, CHANNEL_ACCESS_FAILURE.

— *Loss of acknowledgement*. If a mesh sublayer calls an MAC primitive for data transmission, the execution of the primitive relies on the MAC layer operation. After calling the data request, the mesh sublayer should wait for the confirmation from the MAC layer. If the MAC layer of the transmitter returns MCPS-DATA.confirm within the active duration of the receiver, the mesh sublayer of the transmitter also issues MESH-DATA.confirm with the same status to the next higher layer. However, if the receiver's active or extended active duration expires before finishing the transmission, the mesh sublayer of the transmitter may issue a MCPS-PURGE.request to clear the MAC layer buffet. Otherwise, the MAC layer may issue MCPS-DATA.confirm with NO_ACK status. In either case, the mesh sublayer increases ARC by one and follows retry procedure above.

— *Loss of data*. If a mesh sublayer of the receiver receives an EREQ, it sets a timer that will expire after *meshDATATime*. If the timer is expired before receiving a data frame or another EREQ and the active duration of the receiver ends, the receiver may enter the inactive duration.

Broadcast transmission is started when MESH-DATA.request is issued for broadcast, reliable broadcast, or multicast. If *meshASESOn* is TRUE, the mesh sublayer turns on the receiver and transmits EREQs for the time duration longer than one wakeup interval. The destination address in the mesh payload of EREQ should be 0xffff. Then, the transmitter broadcasts a data frame.

If a device receives a broadcast EREQ from neighbors while broadcasting its own EREQs, it should stop transmitting EREQs and start to receive EREQs to receive a data frame. At this time, the transmitter increases the ASES retry counter, ARC, by one. If the ARC is equal to or greater than *meshMAXNumASESRetries*, the mesh layer issues MESH-DATA.confirm with status, CHANNEL_ACCESS_FAILURE. After receiving the data frame or encountering the *Loss of data* condition defined in 5.5.10.1.3, the transmitter retries ASES broadcasting. If the ARC is less than *meshMAXNumASESRetries*, the mesh layer retries ASES broadcasting.

If the MAC layer confirms with SUCCESS, and it was a broadcast, the mesh sublayer issues MESH-DATA.confirm to the upper layer with SUCCESS. If it was for reliable broadcast or multicast, the mesh sublayer follows the decision from the corresponding transmission method.

Examples of the transmission and message sequence are presented in Table 52 and Figure 53.

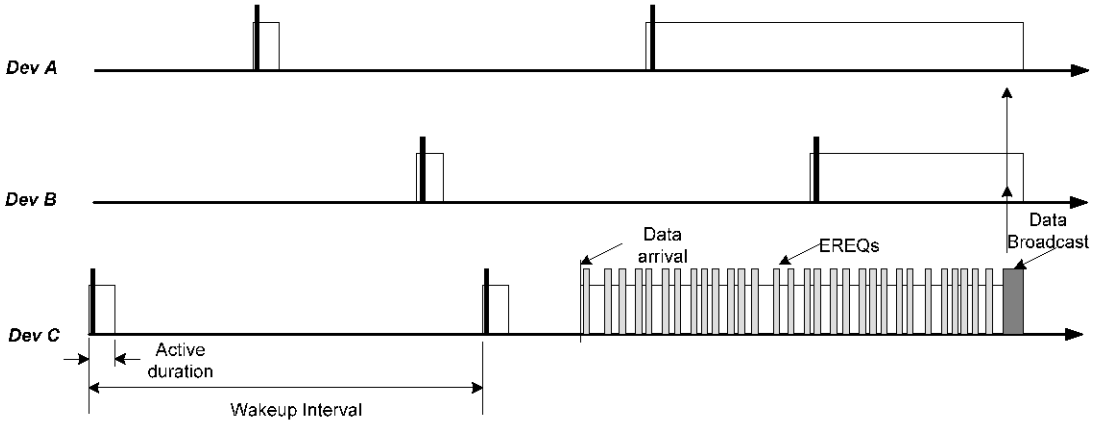

Figure 52—Example of ASES broadcast – Dev C broadcast a frame

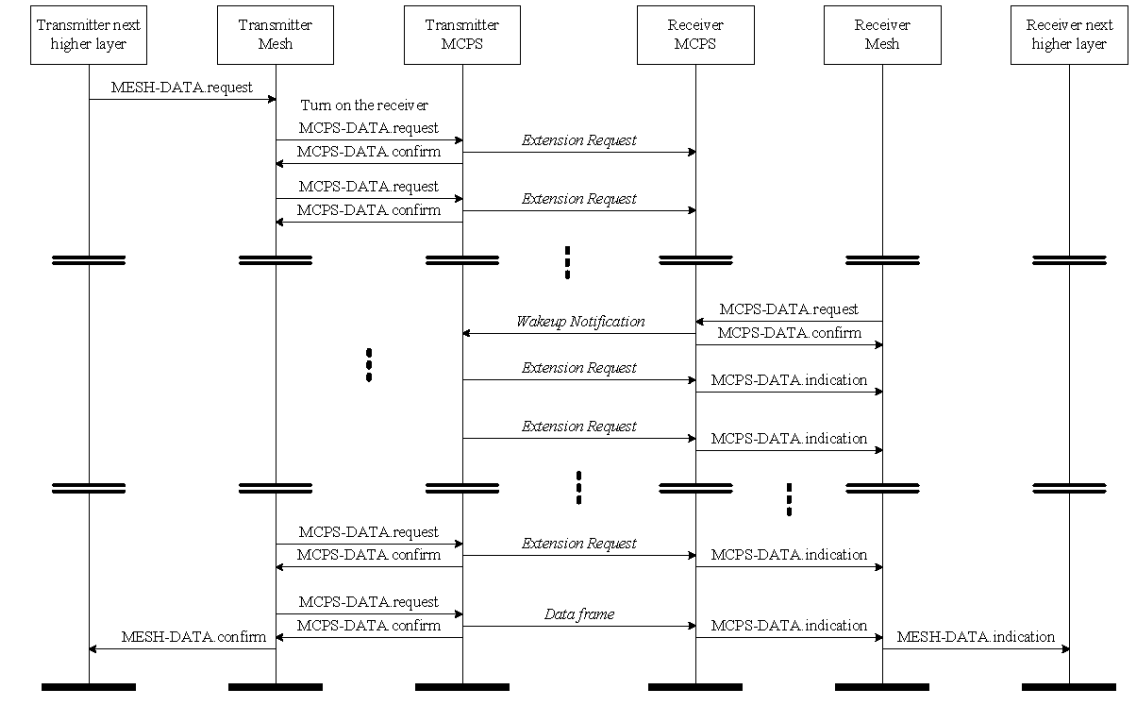

Figure 53—Message sequence chart for ASES broadcast

Command frames may require different transmission methods in ASES. In this subclause, the transmission methods for three kinds of command frames are described.

First, for the frames defined in the IEEE Std 802.15.4-2006, ASES should take different approaches for transmission. There are three categories of exceptional transmission for ASES.

— *Uncontrollable frame transmission*; In IEEE Std 802.15.4-2006, some frames are automatically transmitted at the MAC layer without receiving request primitives from the next higher layer. Thus, these types of frames are not controllable at the mesh sublayer. A beacon frame, a data request frame, an association response frame, a data frame by indirect transmission are in this category. A beacon frame is transmitted after receiving a beacon request frame. Note that the mesh sublayer in this recommended practice does not support the beacon mode of IEEE Std 802.15.4-2006. A beacon is transmitted only upon requested. A data request frame may be transmitted automatically after transmitting an association request frame. A MAC layer sets a timer that will expire after *macResponseWaitTime* then a data request frame is transmitted to receive an association response frame. Thus, this data request for the association response is also uncontrollable. If the data request is transmitted by MLME-POLL.request for other data frames, it may be controllable. Also, an association response frame is uncontrollable because the transmission is automatically performed from the data request frame. Last, all indirect data frames transmitted by a data request frame are uncontrollable due to the same reason. For these uncontrollable frames, the receiver should be ready by turning on the radio as described in 5.5.10.1.4. Note that the mesh sublayer does not support indirect data transmission for a data frame.

— *ASES broadcast emulating transmission*; A beacon request frame is a broadcast frame transmitted by MLME-SCAN.request. This broadcasting may not be received if other devices are in the ASES mode. Thus, the beacon request frame should be transmitted with the same manner as ASES broadcasting although it is not a data frame. For this, a mesh sublayer should transmit EREQs for longer than one wakeup interval and issue MLME-SCAN.request. The number of channels for scaning is restricted to only one to broadcast a stream of EREQs before a beacon request for different channels. In order to scan different channels, the same procedure should be done with a different channel number in the parameter of MLME-SCAN.request.

— *ASES unicast emulating transmssion*; some frames such as an association request frame, a disassociate notification frame, a coordinator realignment frame, and a data request frame (not automatic transmission for association) are the frames defined as unicast in IEEE Std 802.15.4-2006. The frames should be transmitted with the same manner as ASES unicast. Since they require a primitive call before transmission, the transmitter should wait for a WN from the receiver then call the request primitive.

Second, the command frames defined for the ASES should be transmitted as follows:

— *WN transmission*; when a device enters the active duration, a WN is transmitted to notify this state. Before issuing MCPS-DATA.request for a WN transmission, two MAC PIB variables should be adjusted by issuing MLME-SET.request. First, *macMinBE* should be set to one. By doing this, the maximum number of backoff slots before transmitting a WN is constrained to one. Second, *macMaxCSMABackoff* should be set to zero. By doing this, a MAC layer does not retry when the channel is busy.

— *EREQ transmission*; An EREQ is transmitted to extend the active duration of the receiver. Before issuing MCPS-DATA.request for the EREQ, *macMaxCSMABackoff* should be adjusted to zero by issuing MLME-SET.request. By doing this, a MAC layer does not retry when the channel is busy.

— *EREP transmission*; An EREP is transmitted to reply to an EREQ. Before issuing MCPS-DATA.request for the EREP, *macMinBE* and *macMaxCSMABackoff* should be adjusted to two and zero, respectively. By doing this, the MAC layer, with a high probability, attempts to transmit an EREQ. If the channel is busy, it gives up the EREQ transmission.

Last, all other command frames defined in the mesh sublayer should be considered as data frames. Either ASES unicast or ASES broadcast should be used for transmission.

5.5.10.1.4 Initialization

ASES may be started in two ways. First, if *meshASESExpected* is set to FALSE, a device may start or join the network without considering the ASES mode. After establishing a mesh sublayer address, it can start ASES by setting the *meshASESOn* to TRUE. Since all association procedures are performed before starting ASES, the beacon request, the association request, and the data request frame can be transmitted without considering the ASES mode. Also, a beacon frame and an association response frame are received in the same manner.

Second, if *meshASESExpected* is set to FALSE, a device assumes that other devices are running in the ASES mode. The device should follow the sequence below. A device should select one channel before issuing MLME-SCAN.request by triggered by the mesh sublayer MHME-DISCOVER.request, MHME-START-NETWORK.request, and MHME-START-DEVICE.request. Then, it should start with the *ASES broadcasting emulation transmission* described in 5.5.10.1.3. Since the device does not have the mesh address yet, the source address of the EREQs should be set to 0xffff. *ScanDuration* defines how long it has to transmit the EREQs. Note that the *ScanDuration* is defined for the MAC layer timer. The value should be translated to the approximated value in *meshcTimeUnit*. If the mesh sublayer primitive requires more than one channel, it should make a temporary list for the channels to scan. Then, it should scan each channel in the list with the same manner. After the scan procedure, a device should choose one channel to use. This may be done by the mesh sublayer decision or by the parameter in MHME-START-DEVICE.request. The device starts a new mesh or transmits an association request frame with *ASES unicasting emulation transmission* described in 5.5.10.1.3. If the device successfully receives an association response, it should set *meshASESOn* to TRUE. The device starts transmitting WNs and schedules active and inactive durations. *ScanDuration* may be used to set a timer for the WN.

If a device that operates in the ASES mode receives the association request frame, it turns on the receiver circuitry for 2**macResponseWaitTime*. For this exceptional period, the device transmits a WN. All data transmissions follow the same as ASES transmission described in 5.5.10.1.3.

5.5.10.2 Synchronous approach

Synchronous energy saving (SES) is a power saving mechanism which utilizes a synchronous time schedule for data transmission. For SES, all mesh devices should be synchronized network-wide and have the same time structure consisting of an active and an inactive duration.

The features of SES are as follows: First, all the devices should be synchronized to exchange data in the active duration of the time structure at the same time. Then, they are switched to the sleep-mode during the inactive duration of the time structure in order to save energy. Second, the SES uses two types of transmission method: One is the contention-based method by which all of the devices transmit data only in the active duration of time structure through competition; the other is the reservation-based method, by which the devices make a reservation for data transmission in the active duration of the time structure through competition, after which the reserved devices in the inactive duration of the time structure are only allowed to transmit data in the allocated reservation slot duration in order to minimize the delay in data exchange between the end-to-end devices.

5.5.10.2.1 Time structure of SES

A mesh coordinator can optionally organize its time structure for synchronous energy saving. The time structure of synchronous energy saving is described by the values of *meshActiveOrder* and *meshWakeupOrder*. A device may enter an energy saving mode during the inactive duration. The *meshActiveOrder*, a MeshIB attribute, describes the length of active duration. The value of *meshActiveOrder*, AO, and the active duration, AD, are related as follows.

$$AD = meshBaseActiveDuration*2^{AO}\ ,\ 0 \leq AO \leq WO \leq 14$$

The *meshWakeupOrder*, a MeshIB attribute, describes the interval of the time structure. The value of *meshWakeupOrder*, WO, and the wakeup interval are related as follows. The value of *meshWakeupOrder* should be ignored if WO is 15.

$$WI = meshBaseActiveDuration*2^{WO}\ ,\ 0 \leq WO \leq 14.$$

The *meshActiveOrder* and the *meshWakeupOrder* for the SES mode are the same one as in the AES mode. In order to reduce network-wide synchronization errors and to support the scalability of the synchronized network, it is necessary to limit the scope of synchronization. The mesh network can be divided into several synchronized regions for SES. The region synchronizer is a mesh node which is located in each synchronization region and has the responsibility of starting synchronization for its region. To overcome the different time structure, the region synchronizers have to be synchronized with their parent devices and begin to synchronize their child nodes during the next wakeup interval.

A time interval of synchronization should be provided to maintain periodic synchronization, and should be defined as the value of *meshSyncInterval*. A sequential time structure by synchronization region is provided in order to maintain network-wide synchronization.

The inactive duration is temporarily suspended for the period WI for synchronization, and the entire wakeup interval duration is extended to and used in the active duration only to allow all of the devices within the synchronization duration to complete the synchronization procedures. This specific WI for the synchronization is defined as synchronization duration. Please note that the transmission of common data is prohibited within this synchronization duration.

The time structure is shown in Figure 54.

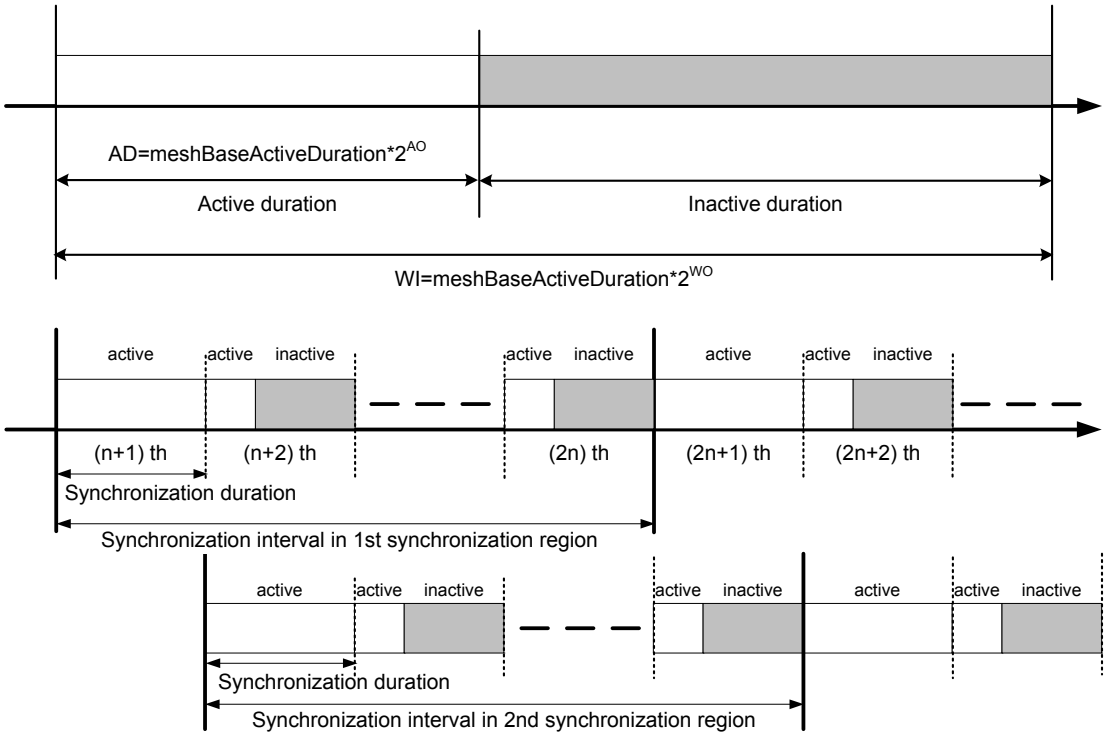

Figure 54—Time structure of SES

5.5.10.2.2 Synchronization algorithm

The synchronization algorithm for SES is a mechanism that all the devices in the network synchronize with the time of a mesh coordinator. Every child device synchronizes with a parent device by receiving two successive synchronization request frames.

Figure 55 illustrates the synchronization algorithm in each step.

Parent Device A broadcasts the first synchronization request command frame to its own child Devices B, C, and D by making use of MCPS-DATA.request primitive. The timestamp value and the synchronization request frame order value of this first synchronization request frame should be set to zero. Once transmission of the synchronization request frame has been successfully performed, the parent device stores the called time of MCPS-DATA.confirm primitive into the value of *meshFirstTxSyncTime* to minimize the delay in the transmission time consumed at the MAC sub-layer. Child devices B, C and D store the called time of MCPS-DATA.indication primitive into the value of *meshFirstRxSyncTime* [step (1)].

The second synchronization request frame of parent device A is transmitted by the same method used for the first synchronization request frame. In such a case, the timestamp value of the synchronization request frame should be set to the *meshFirstTxSyncTime* value of parent device A and the synchronization frame order value of the synchronization specification field should be set to one. Child Devices B, C, and D store the called time of MCPS-DATA.indication primitive into the value of *meshSecendRxSyncTime* [step (2)].

Each child device is synchronized with the time of the parent device by making use of a *meshFirstRxSyncTime* value, a *meshSecondRxSyncTime* and the timestamp value of the second synchronization request frame. The equation for the synchronization of Device B at the time of $T_{current}$ is as follows when the values of the following are provided: the transmission time of the first synchronization request frame of device A (T_{a1}), the receiving time of the first synchronization request frame of Device B (T_{b1}), the receiving time of the second synchronization request frame of Device B (T_{b2}), and the synchronization setup time of device B ($T_{current}$) [step (3)].

$$T_{current} = T_{a1} + (T_{b2} - T_{b1})$$

After child devices are synchronized with the time of the parent device, they transmit the synchronization reply frame to parent device A by making use of CSMA-CA mechanism. Parent device A indicates whether or not its own child devices transmit the synchronization reply frame on the synchronization field of the neighbor list. If receipt of the synchronization reply frame from a specified device fails, the parent device retransmits the synchronization request frame to maintain synchronization [Step (4)].

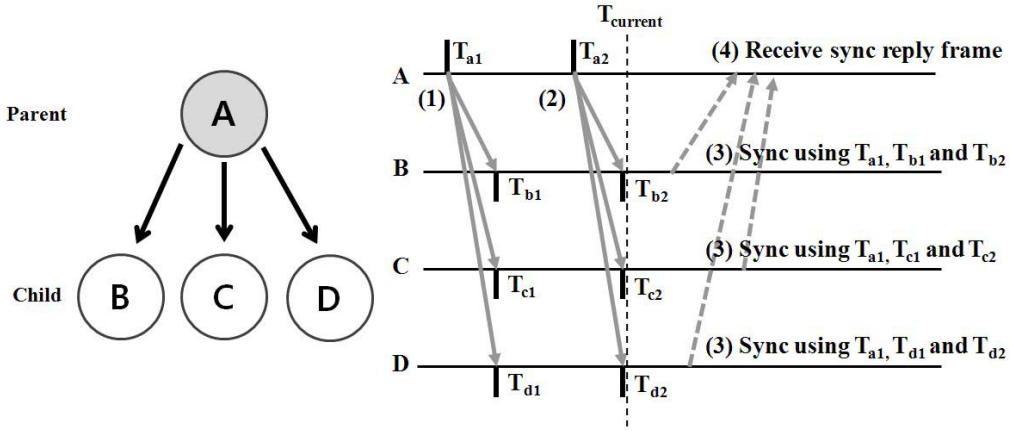

Figure 55—Basic scheme of synchronization

5.5.10.2.3 Start, maintenance and error recovery of synchronization

Synchronization for supporting SES is commenced when the next higher layer of the mesh coordinator calls MESH-START-SYNC.request primitive once the mesh network has been initialized. The mesh sublayer called MESH-START-SYNC.request primitive sets the values of the synchronization specification subfield and the timestamp subfields of the two sequential synchronization request frames, and then transmits the frames to its own child devices. The child devices receiving two synchronization request frames configure the SES time structure and are synchronized to reduce power consumption. The mesh coordinator receives the synchronization reply frames from the child devices in order to manage the synchronization status. Upon receiving the synchronization reply frames from all of the devices, the mesh coordinator notifies this to the next higher layer with MESH-START-SYNC.confirm primitive. If the mesh coordinator fails to receive the frames from any of the child devices, it retransmits the synchronization request frame to the relevant child devices so as to repeat the synchronization procedures. Upon successfully synchronizing with the parent device and having their own child devices, the child devices will repeat the synchronization procedures described above. This process is carried out throughout the network and eventually every node is synchronized to the mesh coordinator device. The entire mesh network is divided into multiple regions along the logical tree and the network-wide synchronization is achieved step-by-step starting from the mesh coordinator and region synchronizer.

All of the devices on the network should be synchronized at every synchronization interval. The mesh coordinator sets the synchronization interval to indicate the cycle, includes it in the synchronization request frame, and informs all of the devices of the cycle. The synchronization interval value is indicated as n*WI, and this value may be determined by taking into account the size of the network and the synchronization error resolution.

The topology throughout the network is subject to the sequential execution of synchronization procedures in each region, which are divided into defined hop distance by the mesh coordinator. The mesh coordinators sets the value of the synchronization region and hop count, includes the value in the synchronization request command frames, and then transmits it to the devices in order to guarantee the synchronization of all of the devices within each synchronization duration for the period WI. If any device receives synchronization request frame, it should check the value of the synchronization region and hop count. If the two values are the same, it means that the receiving device is a region synchronizer device. Otherwise, the receiving device will increment the hop count value by one. This allows the completion of the synchronization procedures within the limited time structure for the network-wide synchronization of large networks.

Figure 56 illustrates the concept of this synchronization region. Device A, a mesh coordinator, makes use of the synchronization request frames within its own synchronization region to start the synchronization procedures at every synchronization interval. These synchronization procedures are executed only in the regions up to the devices within 2-hop distance (Devices A, B, C, D, and E) for the synchronization duration. In the time after the synchronization duration, devices D and E start the synchronization procedures up to the devices within 2-hop distance (Devices D, E, F, G, H, I, J, K, L, and M) from region synchronizer.

Synchronization error recovery procedures are executed for devices that exceed the *meshMaxLostSynchronization* value when they fail to receive the synchronization frames from their parent devices at every synchronization interval, and are identified as no longer being capable of receiving synchronization information from the parent devices.

When any devices that no longer have valid synchronization with the parent devices successfully receive two synchronization request frames from a neighbor device within the next synchronization duration, it should be synchronized with the neighbor device which transmitted synchronization request frame. Then, the device transmits a synchronization reply frame to its neighbor. The neighbor device which transmitted the synchronization request frame will be a sync-parent (Table 46) and the device which transmitted the synchronization reply frame will be a sync-child (Table 46).

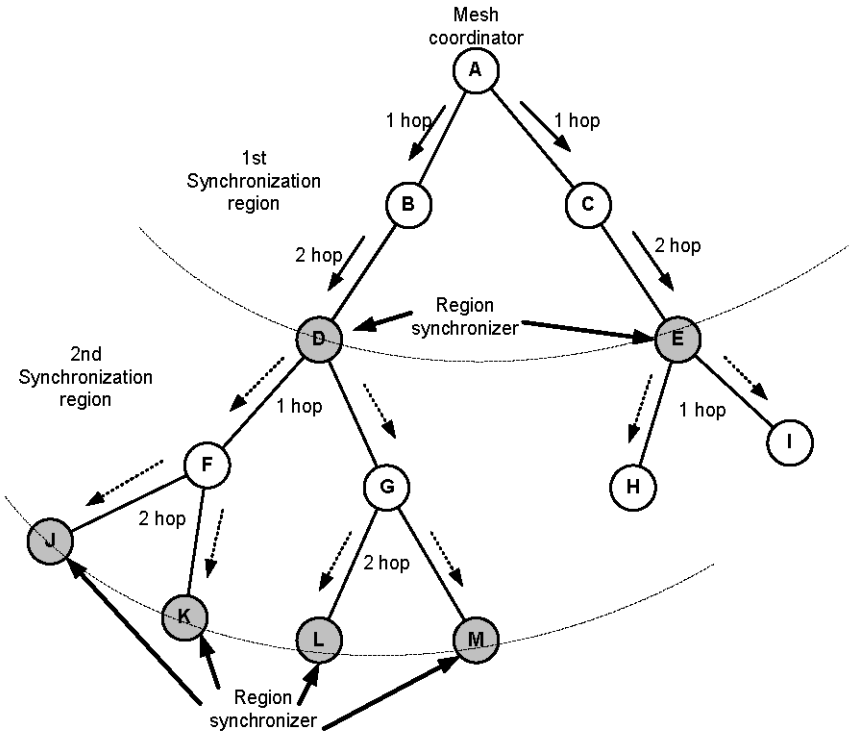

Figure 56—An example that the synchronization region value is 2

5.5.10.2.4 Initialization

When any new devices desire to participate in the network of the SES mode, the next higher layer of devices should first call MHME-DISCOVER.request primitive in order to obtain the network descriptor information from the neighbor devices. In such a case, the scan duration value of MLME-SCAN.request primitive should be set to the minimal value, and the mesh sublayer should repeatedly make calls MLME-SCAN.request to obtain the beacon frames from the devices that operate in the SES mode. Initialization is executed using one of the two methods by verifying the information of the beacon payload whether the current network is operating in the SES mode or not.

First, if the value of SyncEnergySaving in the beacon payload is FALSE, or if the value of SyncEnergySaving is TRUE and the value of WO is 15, initialization is executed in the normal way.

Second, if the value of SyncEnergySaving in the beacon payload is TRUE and the value of WO is not 15, it is allowed to participate in the network by calling MHME-JOIN.request primitive in the active duration by making use of the values of AO and WO of the network description, since the network currently operates in the SES mode. Upon successfully participating in the network with the allocated address, the devices are then synchronized with the parent device after receiving the synchronization request frames from the parent device and operate in the SES mode.

5.5.10.2.5 Data transmission in SES mode

All devices operating in the SES mode should be able to acquire synchronization in order to transmit a frame. There are two variables for supporting SES: *meshSESExpected* and *meshSESOn*. If both values are TRUE, a device assumes that the network is running in the SES mode and has the capability of SES. Otherwise, the network is not permitted to be operated in the SES mode. There are two methods for

transmitting a frame in the SES mode: namely the contention-based method and the reservation-based method.

The contention-based method is that every device transmits frame by making use of CSMA-CA only in the active duration and it is switched to the sleep mode for reducing energy consumption in the inactive period at the same time. This method is used to transmit the data frame of one-hop unicast, multicast, broadcast and reliable broadcast.

Figure 57 illustrates the data transmission of Device A via the contention-based method along with the data transmission route consisting of Devices B, C, and D. When the next higher layer of Device A calls the MESH-DATA.request primitive, the mesh sub-layer will try to transmit a frame to Device B in the active duration. If this attempt succeeds, the mesh sublayer of Device A issues the MESH-DATA.confirm primitive with the status of SUCCESS. The receiving Device B will try to transmit a frame to Device C in the active duration. At this time, the mesh sub-layer will ensure that the remaining operation of transmission can be undertaken and the transaction can be transmitted before the end of the active duration. If the frame was transmitted to Device D, the final destination Device D issues the MESH-DATA.indication primitive to the next higher layer.

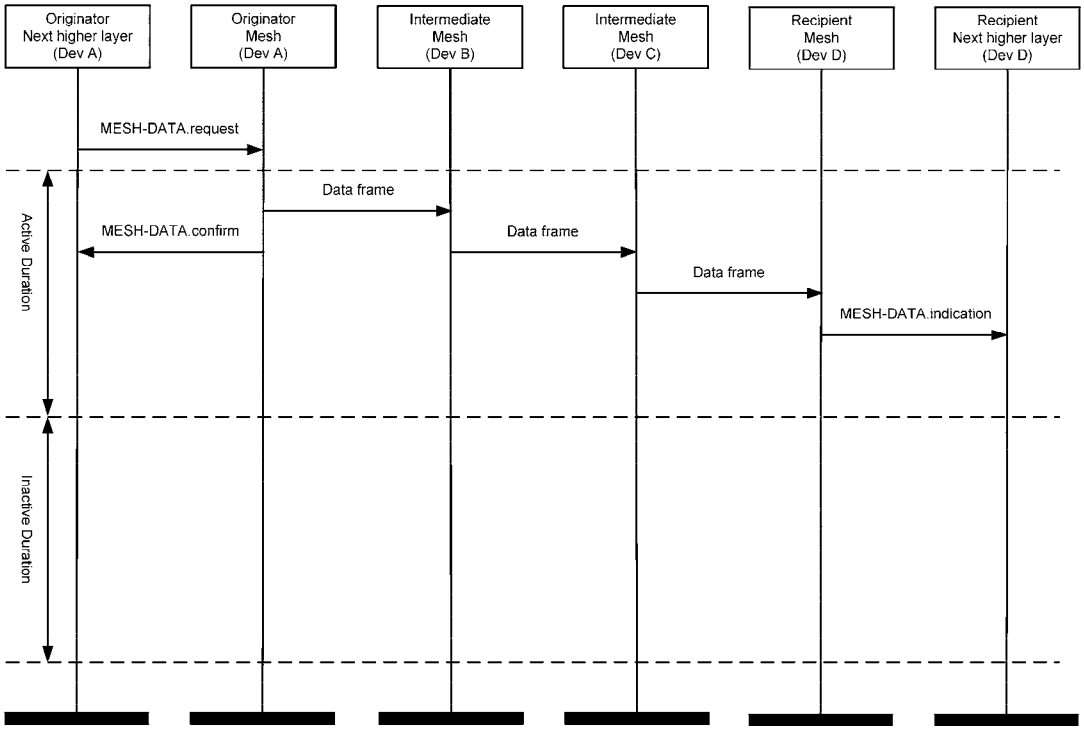

Figure 57—Message sequence chart of contention based method in SES mode

The reservation based method first completes reservation by transmitting a reservation request frame for data transmission in the active duration and then actually transmits data at the reserved time slot in the inactive duration. Since the destination address of the reservation request command is the broadcast address, all of the neighbor devices receive these frames, check the three addresses, the previous address, the next address and the end address, and then take action to continuously proceed with the reservation or to drop the frame. Furthermore, this reservation should be completed in the active duration only. If there is a remaining data transmission, it restarts reservation from the remaining routes in the next active duration. The reason for these procedures is that the reservation of data transmission is not made up to the destination device from the source device; rather, reservation is made at the respective active durations within a certain

route for data transmission. If a device does not receive a reservation request frame or a reservation reply frame up to the active duration, the transmission of requested frame should be stopped. This method is only used to transmit the data frame of multi-hop unicast.

Figure 58 and Figure 59 illustrates the data transmission of Device A via the reservation based method along with the data transmission route consisting of Devices B, C, and D.

Device A makes a request for a reservation by transmitting a reservation request frame in the active duration in order to transmit data to the destination Device D. The three addresses in the payload - the previous address, the next address and the end address - should be set to the address of Devices A, B, and D respectively and the reservation slot for the actual transmission data should be set to zero. Since the destination address in the header of the reservation request frame is 0xffff, all of the neighbor devices receive the data, and the devices check the previous address, next address and end address accordingly. Of the neighbor devices, any device with an address that differs from the next address drops the reservation request frame.

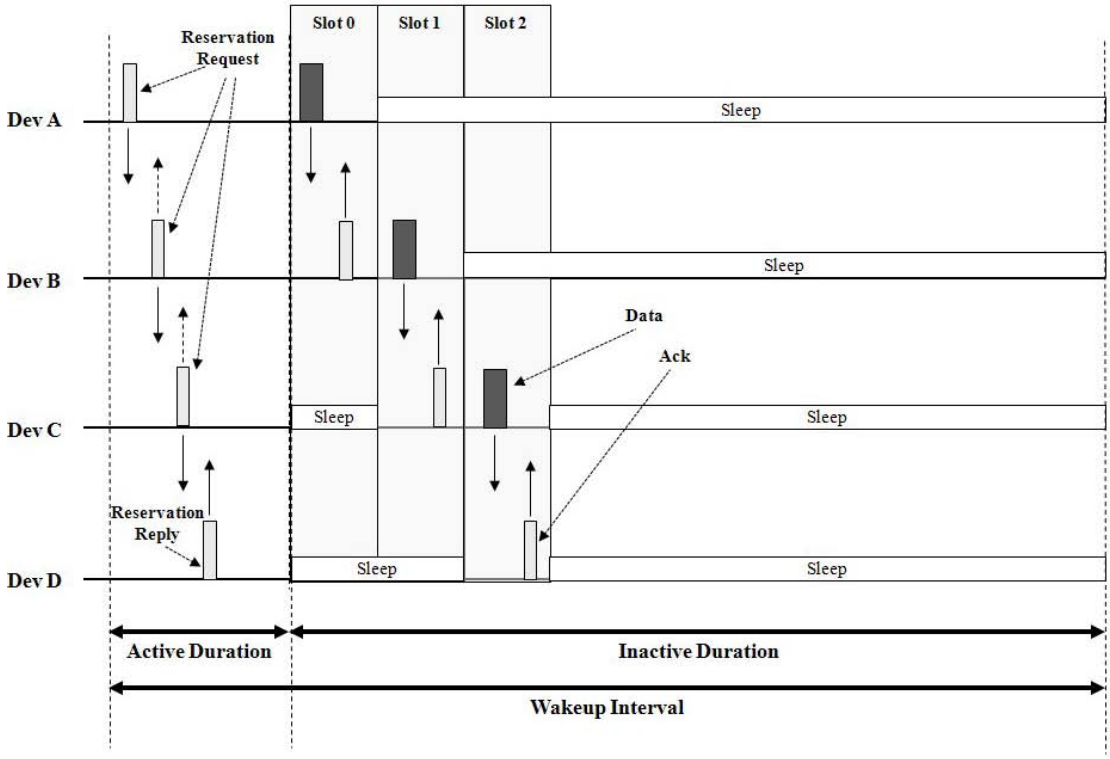

Figure 58—Example of reservation based method in SES mode

Since Device B is the next address but not the end address, the Device B continues to request reservation by transmitting the reservation request frame for the next-hop device whose end address is the final destination. In such a case, the previous, the next and the end addresses should be set to the address of Devices A, C, and D respectively and the reservation slot number should be set to 1. Of the devices receiving the reservation request frames from Device B, Device A, which previously made a request for reservation, can indicates the previous request was successful and prepares data transmission at the reservation slot 0 in the inactive duration, since the previous address is the address of Device A. Of the devices receiving the reservation request frames from Device B, Device C (with an address identical to the next address) processes reservation, while the remaining devices drop the frames.

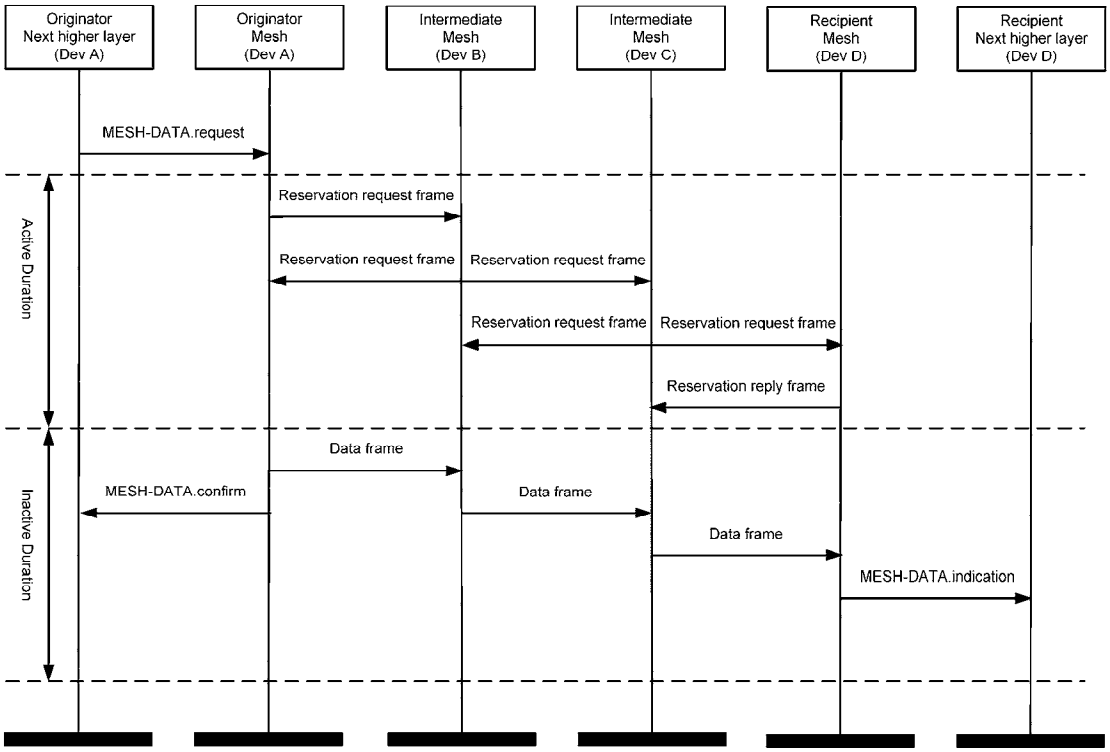

Figure 59—Message sequence chart of reservation based method in SES mode

Device C continues to make requests for reservation as done previously by Device B, since the end address is not the address of the Device C. The previous, next and end addresses of the reservation request frame should be set to the address of Devices B, D, and D respectively and the reservation slot value should be set to 2. Of the devices receiving the reservation request frames from Device C, Device B — with an address identical to the previous address — verifies the completion of reservation, receives data from reservation slot 0 in the inactive duration, and will transmit data at reservation slot 1. Of the devices receiving the reservation request frames from Device C, Device D — with an address identical to the next and the end addresses — transmits the reservation reply frame to Device C to notify the completion of reservation.

Devices A, B, C, and D transmit and receive data according to a sequential basis in the inactive duration to complete data transmission via the reservation based method in the SES mode.

5.5.11 Portability support

This subclause describes the procedure required to support the portability of leaf devices in a LR-WPAN mesh network. The portability support for devices other than leaf devices is beyond the scope of this recommended practice. The portability feature of leaf devices (or, portable devices) is useful in many scenarios for WPAN mesh networks. For example, a multi-function remote control with mesh capability can be brought from room to room to control TVs or other home appliances in a home environment. The procedure for the support of a portable device includes the following components: the detection of the movement, the portable device rejoining the network, the new address assignment, and the notification of the movement to relevant devices.

5.5.11.1 Detection of the movement of a portable device

The movement of a portable device can be detected by either the portable device or the parent of a portable device. When a portable device moves away from its parent, it recognizes the movement when it cannot transmit any packets to its parent device. Also, if the periodic hello command frame, which is an optional feature of this recommended practice, is used in a network, the portable device can detect the movement when it misses the hello command frames from its parent. The portable device's parent can also detect the movement if it misses the MAC layer acknowledgement message from the portable device after sending it a data frame. Whenever a device detects a broken link, its mesh sublayer should react as described in 5.5.6. If the device detects the broken link is a portable device, it can assume it has moved away from its previous location.

5.5.11.2 Rejoining a portable device to the network

After a portable device detected its movement, it should rejoin the network by issuing a MHME-JOIN.request to its mesh sublayer with the parameter RejoinNetwork set to 0x02, which means the device is joining the network using mesh rejoin procedure.

The MHME-JOIN.request will generate an active scan at the MAC layer. Any potential parents receiving this active scan should respond with beacon frames. In each beacon frame, a potential parent should insert its capability to indicate whether it can accept the portable device as a new child. The capability of a device is decided by an upper layer that is beyond the scope of this document.

Upon the reception of beacon frames from potential parents, the portable device will select a "suitable" parent device to join as described in 5.5.2. The portable device should then issue a MLME-ASSOCIATE.request to perform the association procedure at the MAC layer.

The parent device that receives this association, request will first translate the 64-bit IEEE address into a 16-bit logical address using the address matching table as defined in Table 75 and check if this portable device is in its neighbor list. If the portable device is in its neighbor list but with an inactive status (e.g. unknown or down), the parent should assign its old short address to this portable device. Otherwise the parent should assign a new short address selected from its available addressing block. This new short address should be assigned using MLME-ASSOCIATE.response.

After the parent device issues the MLME-ASSOCIATE.response to the portable device, it generates MHME-JOIN.Indication to the upper layer to inform the joining of the portable device. Upon receiving MLME-ASSOCIATE.response, the portable device should generate a MHME-JOIN.confirm to inform its next higher layer the result of the joining process.

5.5.11.3 Informing the movement of the portable device to relevant devices

After successfully re-joined the network, the portable device should report its new address to relevant devices. The relevant devices are defined as the portable device's old parent and devices that have the portable device in their neighbor list. First, it should unicast the rejoin notification command to its former parent. The command frame identifier of rejoin notification should have a value of 0x18. The destination address field is set to former parent's short address. Upon receiving the rejoin notification command, the former parent of the portable device should broadcast the rejoin notification to (TTLOfHello+1) hop-neighbors. Each device has a neighbor list that contains neighbors up to *meshTTLOfHello*+1 hops. Therefore, the old parent needs to broadcast to (*meshTTLOfHello*+1) hop neighbors. Upon receiving the broadcast rejoin notification, every device that has the portable device in its neighbor table will set the status of the portable device to "removed" and update its connectivity matrix. Figure 60 shows this process.

As an example, suppose there is a sender device sending a packet to the portable device using its previous short address. Once the packet arrives at any device that has the address of the portable device in its neighbor list with status "removed", the device should send the rejoin notification with its address field set

to the portable device's new address to the sender and re-route the packet to the portable device's new address as shown in Figure 61.

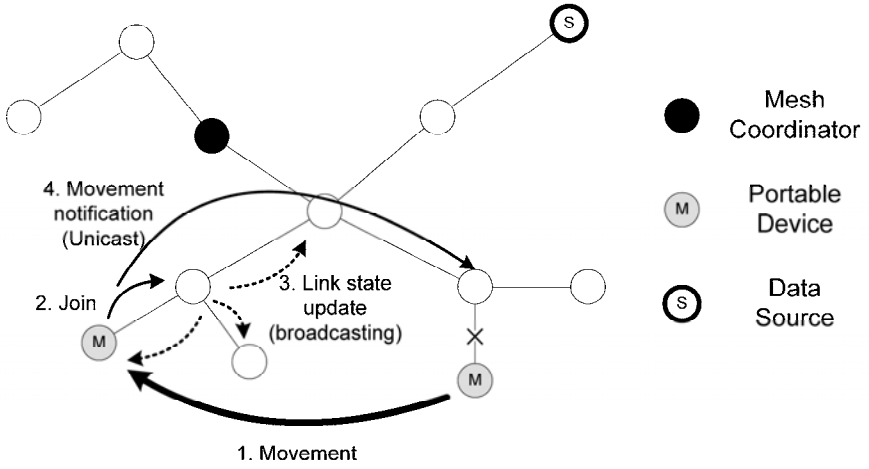

Figure 60—A portable device moves away and informs relevant devices

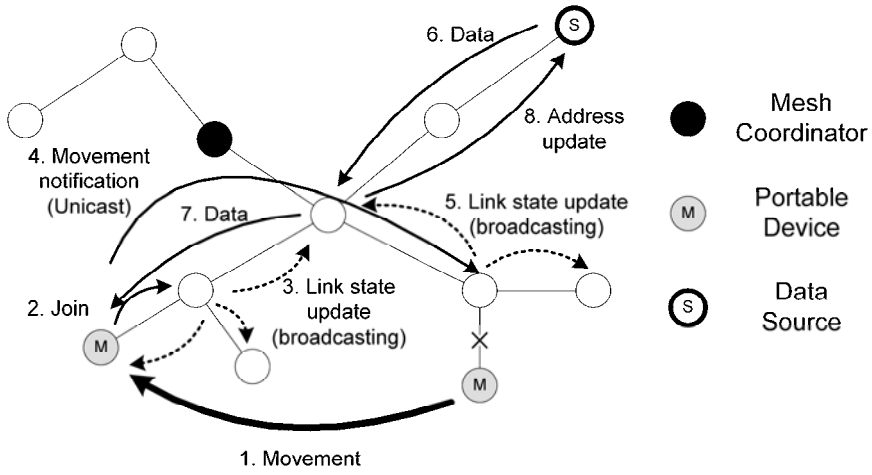

Figure 61— A sender sends data to the portable device

5.5.11.4 Update link state

After the portable device rejoined the network and informed all relevant devices via its old parent of its movement, it should broadcast the hello command frame to *meshTTLOfHello*-hop neighbors in order to inform its new neighbors its address and connectivity.

5.5.12 Traceroute

Traceroute is a service for monitoring the status of a route. It shows the route over the network between two mesh devices, listing all the intermediate routers to its destination. It is helpful to determine whether connections to a given mesh device is good or not, and to figure out where exactly the problem is. The implementation of the traceroute function is optional in this version of the recommended practice.

Traceroute works by sending a series of command frames to the destination with increased TTL values. The first batch of traceroute requests have TTL value of one, indicating only the direct next hop should respond with traceroute reply frames. The next batch of request frames has TTL values of two. This process repeats until the requests reach the final destination.

Figure 62 is an illustration of a three-hop example with *BatchSize* value of two. First, the next high layer of the originator starts its timer *ResponseTimeout* and issues a MHME-TRACE-ROUTE.request primitive to the mesh sublayer. Receiving it, the mesh sublayer transmits out traceroute request frame with TTL value of one. The first intermediate device checks TTL value in the request frame, and transmits back traceroute reply frame to the originator. The traceroute request frame is then discarded. If the sequence number in the reply frame is expected, the mesh sublayer of the originator issues a MHME-TRACE-ROUTE.indication to the next higher layer. The same procedure is repeated for *BatchSize* times, which, in this scenario, equals to two. The indication provides *RTTStamp* which is a round trip time to the intermediate mesh devices or the finial device. *RTTStamp* should be measured in milliseconds from the transmission time of the traceroute request frame to the reception time of the traceroute reply frame. Once a traceroute request frame is transmitted, the originator waits for the traceroute reply frame for *ResponseTimeout.* When the timer expires and a reply has not been received, the originator issues a MHME-TRACE-ROUTE.indication to the next higher layer with the status set to TRACEROUTE_TIMEOUT. Regardless the expected traceroute reply is received or not, the originator transmits the next traceroute request frame until the *Batchsize is* reached.

When a frame passes through mesh devices on the route, the intermediate mesh device decrements the TTL value by one, and forwards the frame to the next mesh device. Intermediate devices should generate the traceroute reply when they find the TTL equals to zero after being decremented by one. In the above example, intermediate2 should find the TTL of the second batch of traceroute request frames equals to two after its own decrement. Therefore it should transmit traceroute reply frames to the originator. The third batch of request frames has TTL values of three and reaches the final destination. The process stops when the originator detects that the last batch of traceroute request frames are sent by the destination. Therefore, after receiving all replies from the destination, the mesh sublayer of the originator issues MHME-TRACE-ROUTE.confirm.

The traceroute confirm returns with DESTINATION_UNREACHABLE in two cases. The first case is when the hop distance to the final destination is farther than *maxTTL*. The other case is when the number of timeout by ResponesTimeout is equal to *BathSize*.

Three parameter values BatchSize, ResponseTimeout, and maxTTL depend upon requirement of the originator’s next higher layers.

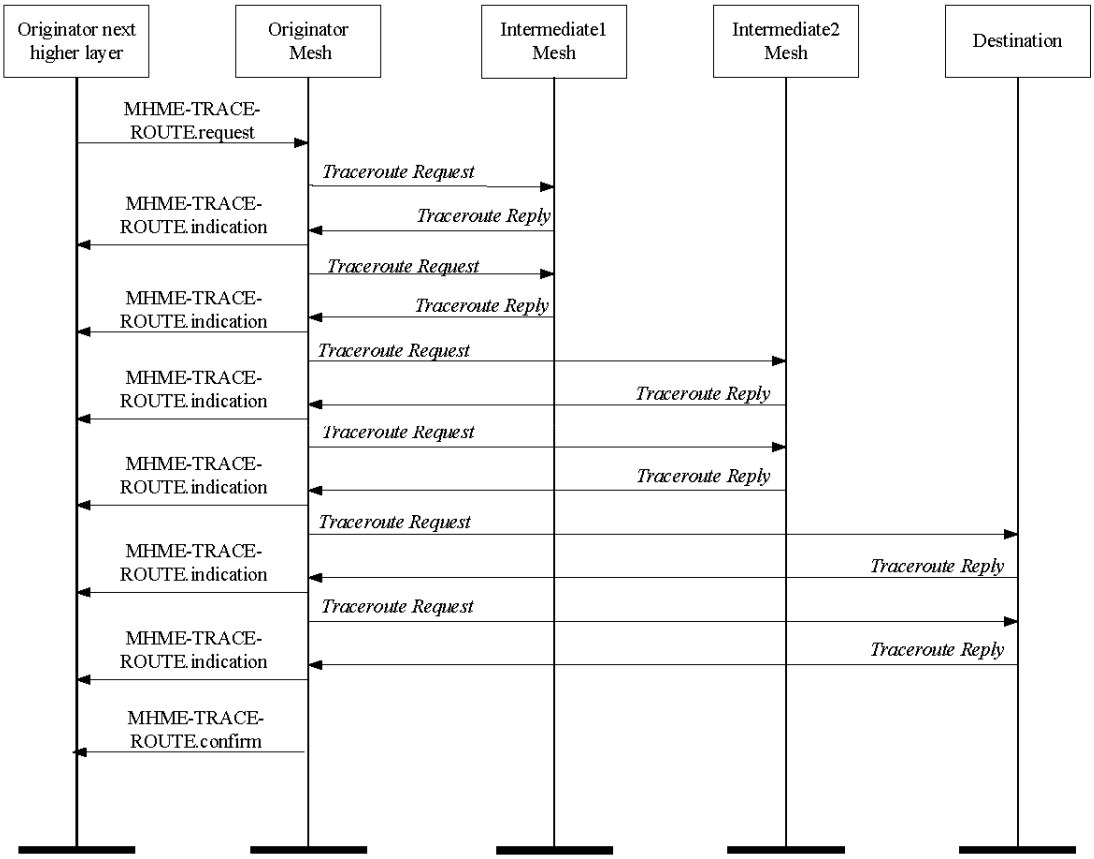

Figure 62—Traceroute procedure

6. High-rate WPAN mesh

6.1 General description

The wireless personal area network (WPAN) is a technology that enables ad hoc connectivity among portable consumer electronics and communications devices. The IEEE Std 802.15.3-2003 and IEEE Std 802.15.3b-2005 have been developed to support emerging multimedia applications such as videoconferencing and HDTV in home environment. The high-rate WPAN mesh is designed to provide network range extension, reliable communication, and efficient bandwidth reuse in high-rate multimedia applications.

This subclause provides an architectural framework that enables high-rate MPNCs to make up a mesh network. The reference model of high-rate WPAN mesh is shown in Figure 63.

Supported features for HR-WPAN include the following:

— network initialization and termination;

— topology formation;

— unicast addressing and routing algorithms.

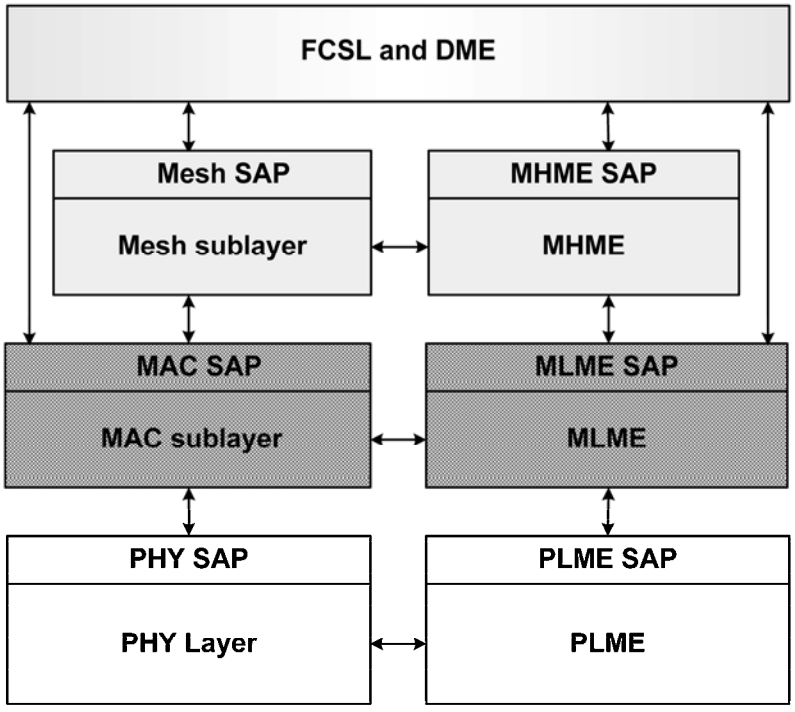

Figure 63—The reference model for high-rate WPAN mesh

In the data plane, the mesh sublayer resides between the frame convergence sublayer (FCSL) and MAC sublayer of IEEE Std 802.15.3-2003 and IEEE Std 802.15.3b-2005. The mesh sublayer provides services to the FCSL via mesh service access point (Mesh SAP).

In the management plane, the mesh sublayer management entity (MHME) resides between the device management entity (DME) and MAC sublayer management entity (MLME) of IEEE Std 802.15.3-2003 and IEEE Std 802.15.3b-2005. The mesh sublayer management entity provides services to the DME via mesh sublayer management entity service access point (MHME SAP).

6.2 MHME SAP interface

The services provided by the MHME to the DME are specified in this subclause. The DME uses the services provided by the MHME through the MHME SAP. The primitives are summarized in Table 52.

Table 52—Summary of MESH primitives

Name	Request	Confirm	Indication	Response
MHME-SET	6.2.1.1	6.2.1.2	–	–
MHME-GET	6.2.1.3	6.2.1.4	–	–
MHME-RESET	6.2.2.1	6.2.2.2	–	–
MHME-SCAN	6.2.3.1	6.2.3.2	6.2.3.3	–
MHME-START	6.2.4.1	6.2.4.2	–	–
MHME-STOP	6.2.5.1	6.2.5.2	–	–
MHME-ASSOCIATE	6.2.6.1	6.2.6.2	6.2.6.3	–
MHME-DISASSOCIATE	6.2.7.1	6.2.7.2	6.2.7.3	–
MHME-TREEID-ASSIGN	0	6.2.8.2	6.2.8.3	–

6.2.1 Generic management primitives

The management information specific to each layer is represented as a personal area network (PAN) information base (PIB).

The MHME, MLME, and PLME are viewed as “containing” the PIB for that layer or sublayer. The generic model of PIB-related management primitives exchanged across the management SAPs is to allow the SAP user entity to either “GET” the value of a PIB attribute, or to “SET” the value of a PIB attribute. The invocation of a SET.request primitive may require the layer entity to perform certain defined actions.

An attempt to “GET” a PIB attribute identified as “write only” attribute is not a valid operation. An attempt to “SET” a PIB attribute identified as a “read only” attribute is not a valid operation.

The GET and SET primitives are represented as requests with associated confirm primitives. The DME uses the services provided by the MHME through the MHME SAP to access the management objects in the mesh PIB, the MAC PIB, or the PHY PIB.

The parameters used for these primitives are defined in Table 53.

Table 53 — MHME generic management primitive parameters

Name	Type	Valid range	Description
PIBattribute	Octet string	Any PIB attribute as defined in 6.4, 6.2.4 or 6.5 of IEEE Std 802.15.3b-2005 or 11.7 of IEEE Std 802.15.3b-2005	The name of the PIB attribute
PIBvalue	Variable	As defined in 6.4, 6.2.4 or 6.5 of IEEE Std 802.15.3b-2005or 11.7 of IEEE Std 802.15.3b-2005.	The PIB value
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the MHME request
ReasonCode	Enumeration	INVALID_ATTRIBUTE_NAME, INVALID_ATTRIBUTE_VALUE, READ_ONLY_ATTRIBUTE, WRITE_ONLY_ATTRIBUTE	Indicates the reason for a ResultCode of FAILURE

6.2.1.1 MHME-SET.request

The primitive attempts to set the indicated PIB attribute of mesh sublayer, MAC sublayer, and PHY layer to the given value. The semantics of this primitive are:

```
MHME-SET.request    (
                    PIBattribute,
                    PIBvalue
                    )
```

The primitive parameters are defined in Table 53.

6.2.1.2 MHME-SET.confirm

The primitive reports the results of an attempt to set the value of a PIB attribute of mesh sublayer, MAC sublayer, and PHY layer. The semantics of this primitive are:

```
MHME-SET.confirm    (
                    PIBattribute,
                    ResultCode,
                    ReasonCode
                    )
```

The primitive parameters are defined in Table 53.

6.2.1.3 MHME-GET.request

The primitive requests information about a given PIB attribute. The semantics of this primitive are:

```
MHME-GET.request    (
                    PIBattribute
                    )
```

The primitive parameter is defined in Table 53.

6.2.1.4 MHME-GET.confirm

The primitive reports the results of an information request about a given PIB attribute. The semantics of this primitive are:

```
MHME-GET.confirm    (
                    PIBattribute,
                    PIBvalue,
                    ResultCode,
                    ReasonCode
                    )
```

The primitive parameters are defined in Table 53.

6.2.2 Resetting the mesh sublayer/MHME

These primitives support the process of resetting the mesh sublayer/MHME, MAC sublayer/MLME, and PHY layer/PLME. The parameters used for these primitives are defined in Table 54.

Table 54 — MHME-RESET primitive parameters

Name	Type	Valid range	Description
SetDefaultPIB	Boolean	TRUE, FALSE	If TRUE, all PIB attributes are set to their default values. The default values are implementation-dependent. If FALSE, the mesh entity is reset, but all PIB attributes retain the values that were in place prior to the generation of the MHME-RESET.request primitive.
ResultCode	Enumeration	READY, ERROR	If READY, the MHME has successfully completed initialization and is ready to receive requests. If ERROR, the MHME was unable to successfully complete initialization and is unable to receive requests.

6.2.2.1 MHME-RESET.request

This primitive requests that the entity be reset to its initial conditions. The semantics of this primitive are:

```
MHME-RESET.request  (
                    SetDefaultPIB
                    )
```

The primitive parameter is defined in Table 54.

The MHME restores the mesh sublayer and its internal variables (other than PIB attributes) to their initial state and values. The SetDefaultPIB parameter governs whether PIB attributes are reinitialized or left unchanged.

6.2.2.2 MHME-RESET.confirm

This primitive is generated by the MHME when it completes its initialization process. The semantics of this primitive are:

```
MHME-RESET.confirm      (
                          ResultCode
                        )
```

The primitive parameter is defined in Table 54.

6.2.3 Scanning for mesh networks

These primitives support the process of determining the presence or absence of a mesh network. The parameters used for these primitives are defined in Table 55.

Table 55— MHME-SCAN primitive parameters

Name	Type	Valid range	Description
ScanForBSID	Octet string	TRUE, FALSE	Indicates if the scan process should search for a specific BSID, as described in 8.2.1 of IEEE Std 802.15.3-2003.
BSIDLength	Variable	As defined in 7.4.2 of IEEE Std 802.15.3-2003.	The number of octets in the BSID.
BSID	Enumeration	As defined in 7.4.2 of IEEE Std 802.15.3-2003.	The text string of a specific piconet for which to scan. This parameter is not used if ScanForBSID is FALSE.
ScanForTREEID	Enumeration	TRUE, FALSE	Indicates if the scan process should search for a specific TREEID, as described in 6.6.3.
TREEID	Integer	0–65535	The TREEID of a specific piconet for which to scan. This parameter is not used if ScanForTREEID is FALSE.
ScanForMPNCAddress	Boolean	TRUE, FALSE	Indicates if the scan process should search for an MPNC with a specific MAC address, as described in 8.2.1 of IEEE Std 802.15.3-2003.
MPNCAddress	MAC address	Any valid individual MAC address, as described in 7.1 of IEEE Std 802.15.3-2003.	The MAC address of a specific MPNC for which to scan. This parameter is not used if ScanForPNCAddress is FALSE.

Table 55— MHME-SCAN primitive parameters *(continued)*

Name	Type	Valid range	Description
Timeout	Integer	0–65535	The time in milliseconds allowed for the primitive to complete.
NumberOfPiconets	Integer	0–255	The number of mesh networks found during the scanning process.
PiconetDescriptionSet	N/A	A set containing zero or more instances of a MeshDescription	The PiconetDescriptionSet is returned to indicate the results of the scan request.
NumberOfChannels	Integer	0 to the maximum number of PHY-dependent channels, as defined in 11.2.3 of IEEE Std 802.15.3-2003.	Indicates the number of channels scanned.
ChannelRatingList	Ordered list of integers	0 to the maximum number of PHY-dependent channels, as defined in 11.2.3 of IEEE Std 802.15.3-2003.	Specifies a list of the channels scanned ordered from the best to the worst in terms of interference.
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the MHME request.
ReasonCode	Enumeration	OTHER	Indicates the reason for a ResultCode of FAILURE.

In Table 55, a PiconetDescriptionSet is a set of PiconetDescriptions. Each PiconetDescription consists of the elements shown in Table 56.

Table 56—Elements of PiconetDescription

Name	Type	Valid range	Description
BSIDLength	Integer	As defined in 7.4.2 of IEEE Std 802.15.3-2003	The number of octets in the BSID.
BSID	Octet String	As defined in 7.4.2 of IEEE Std 802.15.3-2003	The text string identifier of a discovered piconet.
PNID	Integer	0–65535	The PNID of a discovered piconet.
MPNCAddress	MAC address	Any valid individual MAC aadress	The MAC address of the MPNC of the piconet that was found.
ChannelIndex	Integer	0 to the maximum number of PHY-dependant channels.	A PHY-dependent channel number on which the piconet was found.
SECmode	Enumeration	MODE_0, MODE_1	The security mode of the piconet that was found, as described in 7.3.1 of IEEE Std 802.15.3-2003.
SignalQuality	Integer	0–15	Indicates the quality of the received frame or beacon for this piconet. The value is implementation-dependent with 0 indicating the lowest quality and 15 the highest quality.
NumApplication SpecificationData	Integer	0–255	Indicates the number of ApplicationSpecificDataSets advertised in the beacon of a discovered piconet.
Application SpecificDataSet	Set of application specific data, as defined in Table 57.	A set containing zero or more instances of ApplicationSpecificData	The ApplicationSpecificDataSet is returned to indicate the ApplicationSpecificData offered by the discovered piconet.

In Table 55, the ChannelRatingList is a set of N integer values, where N equals the number of channel numbers provided by the PHY. The elements of the set are channel numbers and they are ordered from best (least interference) at the lowest set index to worst (most interference) at the highest set index.

In Table 56, an ApplicationSpecificDataSet is a set of ApplicationSpecificData. Each ApplicationSpecificData consists of the elements shown in Table 57. The ApplicationSpecificDataSets are only the current values advertised in the beacon and potentionally are different in subsequent beacons.

Table 57—Elements of ApplicationSpecificData

Name	Type	Valid range	Description
OUI	Octet string	Any valid OUI, as defined in 7.4.7 of IEEE Std 802.15.3b-2005	The unique identifier for the data, as described in 7.4.7 of IEEE Std 802.15.3b-2005
ApplicationDataLength	Integer	As defined in 7.4.7 of IEEE Std 802.15.3b-2005	The length of the ApplicationData in Octets
ApplicationData	Octet string	Any valid octet string of length up to Application-DataLength	Application-dependent data, as described in 7.4.7 of IEEE Std 802.15.3b-2005

Any security features of an existing piconet are ignored during the scan process.

It is not possible to obtain piconet services information during the scan process.

6.2.3.1 MHME-SCAN.request

This primitive is used to initiate the passive scan procedure to search for either a specific mesh network or any mesh network. The semantics of this primitive are:

```
MHME-SCAN.request   (
                        ScanForBSID,
                        BSIDLength,
                        BSID,
                        ScanForTREEID,
                        TREEID,
                        ScanForMPNCAddress,
                        MPNCAddress,
                        Timeout
                        )
```

The primitive parameters are defined in Table 55.

6.2.3.2 MHME-SCAN.confirm

This primitive is used to report the result of the request to initiate the passive scan procedure to search for either a specific mesh network or any mesh network. The semantics of this primitive are:

```
MHME-SCAN.confirm            (
                        NumberOfPiconet,
                        PiconetDescriptionSet,
                        NumberOfChannels,
                        ChannelRatingList,
                        ResultCode,
                        ReasonCode
                        )
```

The primitive parameters are defined in Table 55. All of the mesh networks found during the scan will be reported in separate elements of the MeshDescriptionSet, even if more than one mesh network is found on a given channel.

6.2.3.3 MHME-SCAN.indication

This primitive is used to report the result of a passive scan that was initiated by the mesh sublayer. The semantics of this primitive are:

```
MHME-SCAN.indication        (
                            NumberOfPiconet,
                            PiconetDescriptionSet,
                            NumberOfChannels,
                            ChannelRatingList
                            )
```

The primitive parameters are defined in Table 55.

6.2.4 Starting a mesh-enabled piconet

These primitives support the process of starting a mesh network as a MC or child piconet after an association to the mesh coordinatior. The parameters used for these primitives are defined in Table 58.

Table 58— MHME-START primitive parameters

Name	Type	Valid range	Description
MeshID	Integer	0–65535	The MeshID of the new mesh network.
HopCountToMC	Integer	0–255	Indicates the number of hop count to the MC.
StartingTREEID	Integer	0–65535	The lowest TREEID of the TREEID block, which is assigned by the parent MPNC.
EndingTREEID	Integer	0–65535`	The highest TREEID of the TREEID block, which is assigned by the parent MPNC.
BSIDLength	Integer	As defined in 7.4.2 of IEEE Std 802.15.3-2003	The number of octets in the BSID.
BSID	Octet string	As defined in 7.4.2 of IEEE Std 802.15.3-2003	The BSID of the new piconet.

Table 58— MHME-START primitive parameters *(continued)*

Name	Type	Valid range	Description
SECMode	Enumeration	MODE_0, MODE_1	The security mode of the piconet, as described in 7.3.1 of IEEE Std 802.15.3-2003.
MinDepSuperframePercent	Integer	1–100	The minimum percent of the superframe requested as a CTA for the dependent piconet, as described in 8.2.5 and 8.2.6 of IEEE Std 802.15.3b-2005.
DesiredDepSuperframePercent	Integer	1–100	The desired percent of the superframe requested as a CTA for the dependent piconet, as described in 8.2.5 and 8.2.6 of IEEE Std 802.15.3b-2005.
AllocatedSuperframePercent	Integer	0–100	The percent of the superframe allocated to the new dependent piconet. If the channel time request was rejected, the value should be set to zero.
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the MHME request.
ReasonCode	Enumeration	NOT_MPNC_CAPABLE, NO_CHANNELS_AVAILABLE, ALREADY_MPNC, NOT_MESH_CAPABLE_IE_DELIVERABLE, OTHER	Indicates the reason for a Result-Code of FAILURE.

6.2.4.1 MHME-START.request

This primitive is used to start a mesh network as a MC or child piconet after an association to the mesh coordinatior. The semantics of this primitive are:

```
MHME-START.request          (
                              MeshID,
                              HopCountToMC,
                              StartingTREEID,
                              EndingTREEID,
                              BSIDLength,
                              BSID,
                              SECMode,
                              MinDepSuperframePercent,
                              DesiredDepSuperframePercent
                              )
```

The primitive parameters are defined in Table 58.

6.2.4.2 MHME-START.confirm

This primitive reports the results of a piconet creation procedure. The semantics of this primitive are:

```
MHME-START.confirm          (
                              AllocatedSuperframePercent,
                              ResultCode,
                              ReasonCode
                              )
```

The primitive parameters are defined in Table 58.

6.2.5 Stopping a mesh network

These primitives support the process of stopping operation as an MPNC regardless of if the MPNC is a MC. The process may result shutdown of mesh network or the handover of MC operation to another MPNC in the mesh network. The parameters used for these primitives are defined in Table 59.

Table 59—MHME-STOP primitive parameters

Name	Type	Valid range	Description
RequestType	Enumeration	SHUTDOWN	The current piconet operations will be stopped.
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the MHME request.
ReasonCode	Enumeration	NOT_AN_MPNC, OTHER	Indicates the reason for a ResultCode.

6.2.5.1 MHME-STOP.request

This primitive initiates the mesh network shutdown procedure or the mesh network handover procedure. The semantics of this primitive are:

```
MHME-STOP.request       (
                            RequestType
                        )
```

The primitive parameters are defined in Table 59.

6.2.5.2 MHME-STOP.confirm

This primitive reports the results of the request to stop operation as a MC. The semantics of this primitive are:

```
MHME-STOP.confirm       (
                            ResultCode,
                            ReasonCode
                        )
```

The primitive parameters are defined in Table 59.

6.2.6 Associating with a mesh network

The following primitives support the process of an MPNC associating with a mesh network. The parameters used for these primitives are defined in Table 60.

Table 60— MHME-ASSOCIATE primitive parameters

Name	Type	Valid range	Description
MeshID	Integer	0–65535	The identification of the mesh network
ParentTREEID	Integer	0-65535	The TREEID of the target MPNC for an association process, as described in 6.6.2.
ParentMPNCAddress	MAC Address	Any valid individual MAC address	The MAC address of the target MPNC for the association
BSIDLength	Integer	As defined in 7.4.2 of IEEE Std 802.15.3-2003	The number of octets in the BSID.
BSID	Octet string	As defined in 7.4.2 of IEEE Std 802.15.3-2003	The BSID of the target MPNC for the association.
ChannelIndex	Integer	0–255	A PHY-dependent channel number to search for the target MPNC for the association.

Table 60—MHME-ASSOCIATE primitive parameters *(continued)*

Name	Type	Valid range	Description
PiconetServicesInquiry	Boolean	TRUE, FALSE	Requests that the MPNC send the services information about the piconet, as described in 8.3.2 of IEEE Std 802.15.3b-2005.
DEVID	Integer	Any valid DEVID, as defined in 7.2.3 of IEEE Std 802.15.3b-2005.	The DEVID assigned to an MPNC as the result of an association or the DEVID of an MPNC that has joined the piconet.
DEVAddress	MAC Address	Any valid individual MAC address	The MAC address of an MPNC for that has joined the piconet.
NeighborPiconetRequest	Boolean	TRUE, FALSE	Indicates that the MPNC will join as a neighbor MPNC rather than as a member of the piconet.
VendorSpecificIE	Octet string	Any valid Vendor Specific IE, as defined in 7.4.17 of IEEE Std 802.15.3-2003	The Vendor Specific IE, if present, in the Association Response command, as described in 7.5.1.2 of IEEE Std 802.15.3-2003.
Timeout	Integer	0–65535	The time in milliseconds allowed for the primitive to complete.
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the MHME request.
ReasonCode	Enumeration	REQUEST_TIMEOUT, MPNC_NOT_FOUND, MPNC_DENIED, MPNC_BUSY, ALREADY_ASSOCIATED, OTHER	Indicates the reason for a ResultCode of FAILURE.

6.2.6.1 MHME-ASSOCIATE.request

This primitive initiates the association procedure. The semantics of this primitive are:

```
MHME-ASSOCIATE.request (
                        MeshID,
                        BSIDLength,
                        BSID,
                        ParentTREEID,
                        ParentMPNCAddress,
                        ChannelIndex,
                        NeighborPiconetRequest,
                        PiconetServiceInquiry,
                        Timeout
                        )
```

The primitive parameters are defined in Table 60.

6.2.6.2 MHME-ASSOCIATE.confirm

This primitive reports the result of the association procedure. The semantics of this primitive are:

```
MHME-ASSOCIATE.confirm (
                        DEVID,
                        VendorSpecificIE,
                        ResultCode,
                        ReasonCode
                        )
```

The primitive parameters are defined in Table 60.

6.2.6.3 MHME-ASSOCIATE.indication

The primitive is used to indicate that a new DEV has associated with the same mesh network. The semantics of this primitive are:

```
MHME-ASSOCIATE.indication (
                           DEVID,
                           DEVAddress
                           )
```

The primitive parameters are defined in Table 60.

6.2.7 Disassociation from a mesh network

The following primitives are used when an MPNC disassociates from a mesh network or when MC disassociates an MPNC. The parameters used for these primitives are defined in Table 61.

Table 61—MHME-DISSASOCIATE primitive parameters

Name	Type	Valid range	Description
MeshID	Integer	0–65535	The identification of the mesh network.
TREEID	Integer	0–65535	The TREEID of the target MPNC for the disassociation, as described in 6.6.4.
DEVID	Integer	Any valid DEVID, as defined in 7.2.3 of IEEE Std 802.15.3b-2005.	The DEVID of the disassociated MPNC.
DEVAddress	MAC address	Any valid individual MAC address	The MAC address of the disassociated MPNC.
Timeout	Integer	0–65535	The time in milliseconds allowed for the primitive to complete.
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the MHME request.
ReasonCode	Enumeration	REQUEST_TIMEOUT, NOT_ASSOCIATED, CURRENTLY_MPNC, DEV_ATP_EXPIRED, MPNC_ATP_EXPIRED, OTHER_MPNC_ACTION, UNKNOWN	Indicates the reason for the disassociation of an MPNC from a piconet.

6.2.7.1 MHME-DISASSOCIATE.request

This primitive initiates the procedure for an MPNC to disassociate from a mesh network. The semantics of this primitive are:

```
MHME-DISASSOCIATE.request    (
                             Timeout
                             )
```

The primitive parameter is defined in Table 61.

6.2.7.2 MHME-DISASSOCIATE.confirm

This primitive is used to confirm the result of the disassociation procedure. The semantics of this primitive are:

```
MHME-DISASSOCIATE.confirm    (
                             ResultCode,
                             ReasonCode
                             )
```

The primitive parameters are defined in Table 61.

6.2.7.3 MHME-DISASSOCIATE.indication

This primitive is used to indicate that either this MPNC or another MPNC has been disassociated from the mesh network. The semantics of this primitive are:

```
MHME-DISASSOCIATE.indication    (
                                MeshID,
                                TREEID,
                                DEVID,
                                DEVAddress,
                                ResultCode,
                                ReasonCode
                                )
```

The primitive parameters are defined in Table 61.

6.2.8 TREEID assignment

The following primitives are used when an MPNC requests its own TREEID block from its parent MPNC. The Parent MPNC may or may not be the MC. The MC or the parent MPNC assigns a TREEID block to the child MPNCs. The parameters used for these primitives are defined in Table 62.

Table 62— MHME-TREEID-ASSIGN primitive parameters

Name	Type	Valid range	Description
DEVID	Integer	Any valid DEVID, as defined in 7.2.3 of IEEE Std 802.15.3b-2005	The DEVID of the MPNC requesting TreeID
TrgtTREEID	Integer	0–65535	The TREEID of the target MPNC for an TREEID assignment process, as described in 6.6.3
NumberOfDescendant	Integer	0–255	The desired number of descendants of a MPNC
StartingTREEID	Integer	0–65535	The lowest TREEID of the TREEID block, which is assigned by the parent MPNC
EndingTREEID	Integer	0–65535	The higest TREEID of the TREEID block, which is assigned by the parent MPNC including the MC
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the TREEID request
ReasonCode	Enumeration	NO_TREEID_BLOCK_AVAILABLE, NO_CTA_AVAILABLE, NO_CHILD_MPNC_SUPPORT	Indicates the reason for Indicates the reason for the failure of primitive

6.2.8.1 MHME-TREEID-ASSIGN.request

This primitive requests a TREEID block from the parent MPNC. The semantics of this primitive are:

```
MHME-TREEID-ASSIGN.request      (
                                TrgtTREEID,
                                NumberOfDescendant
                                )
```

The primitive parameter is defined in Table 62.

6.2.8.2 MHME-TREEID-ASSIGN.confirm

This primitive is used to confirm the result of the TREEID assignment procedure. The semantics of this primitive are:

```
MHME-TREEID-ASSIGN.confirm      (
                                StartingTREEID,
                                EndingTREEID,
                                ResultCode,
                                ReasonCode
                                )
```

The primitive parameters are defined in Table 62.

6.2.8.3 MHME-TREEID-ASSIGN.indication

This primitive is used to indicate that a child MPNC wants to obtain a TREEID block from the parent MPNC including an MC. The semantics of this primitive are:

```
MHME-TREEID-ASSIGN.indication   (
                                DEVID,
                                StartingTREEID,
                                EndingTREEID
                                )
```

The primitive parameters are defined in Table 62.

6.3 Mesh SAP interface

The Mesh SAP defines the logical interface between the mesh sublayer and FCSL. This logical interface description includes a list of primitives and their definitions. The Mesh SAP primitives are summarized in Table 99.

Table 63—Summary of Mesh SAP primitives

Name	Request	Confirm	Indication	Response
MESH-ASYNC-DATA	6.3.1	6.3.2	6.3.3	–
MESH-ISOCH-DATA	6.3.4	6.3.5	6.3.6	–

Table 64— MESH-ASYNC-DATA and MESH-ISOCH-DATA Primitive Parameters

Name	Type	Valid range	Description
MeshID	Integer	0–65535	Specifies the identification of a mesh network.
TrgtTREEID	Integer	Any valid PNID, as defined in 7.2.2 of IEEE Std 802.15.3-2003.	Specifies the target TREEID of an MHSDU.
OrigTREEID	Integer	Any valid PNID, as defined in 7.2.2 of IEEE Std 802.15.3-2003.	Specifies the origin TREEID of an MHSDU.
RequestID	Integer	0–511	An identifier, which is used to correlate a request to a response. It is unique per stream index for outstanding isochronous requests. It is unique among all outstanding asynchronous requests.
TransmitTimeout	Duration	$1–(2^{32} – 1)$	Maximum allowed delay in microseconds from when the MSDU is been presented to the MAC SAP until the frame has finished transmission and the acknowledgment, if required is successfully received.
StreamIndex	Integer	Any valid stream index, as defined in 7.2.5 of IEEE Std 802.15.3b-2005	The stream with which the data isassociated.
MaxRetries	Integer	0–255	Specifies the maximum number of retries to attempt per transmitted frame with a maximum value no greater than the MaxRetries value supplied in the original stream creation request.
SECMode	Boolean	TRUE, FALSE	Indicates if security is to be applied to the MSDU or if the MSDU was received securely.
UserPriority	Integer	As defined in Table A.1 of IEEE Std 802.15.3b-2005	User priority of the stream, as described in Table A.1 of IEEE Std 802.15.3b-2005.
ACKRequested	Boolean	TRUE, FALSE	Indicates if the request requires an acknowledgment, as described in 8.8 of IEEE Std 802.15.3b-2005, the MSDU at the MAC sublayer.

Table 64— MESH-ASYNC-DATA and MESH-ISOCH-DATA Primitive Parameters *(continued)*

Name	Type	Valid range	Description
ConfirmRequested	Duration	NEVER, ALWAYS, ON_ERROR	Indicates when a confirm primitive is required for the request.
SNAPHeaderPresent	Boolean	TRUE, FALSE	TRUE indicates that a logical link control/ subnetwork access protocol (LLC/SNAP) header is present in the dataframe.
Length	Integer	0–pMaxTransferUnitSize	The length of the MHSDU in octets.
Data	Variable number of octets	Any octet string the length of which is given by the Length parameter	MHSDU portion of the primitive.
TransmitDelay	Duration	$1–(2^{32} – 1)$	Delay in microseconds from when the MSDU is presented to the Mesh SAP until the frame has finished transmission and the acknowledgment, if required, has been successfully received. If the transmission fails due to timeout, this field should be set to the TransmitTimeout value for this frame.
TrgtID	Integer	Any valid DEVID, asdefined in 7.2.3 of IEEE Std 802.15.3b-2005.	Specifies the target DEVID of an MHSDU.
OrigID	Integer	Any valid DEVID, asdefined in 7.2.3 of IEEE Std 802.15.3b-2005.	Specifies the origin DEVID of an MHSDU.
ResultCode	Enumeration	SUCCESS, FAILURE.	Indicates the result of the MHME request.
ReasonCode	Enumeration	TRANSMIT_TIMEOUT, MAX_RETRIES, NOT_ASSOCIATED, OTHER.	The reason for a ResultCode of FAILURE.

6.3.1 MESH-ASYNC-DATA.request

This primitive is used to initiate the transfer of an MHSDU from one mesh entity to another mesh entity. The semantics of this primitive are:

```
MESH-ASYNC-DATA.request (
                        RequestID,
                        MeshID,
                        TrgtTREEID,
                        TrgtID,
                        TransmitTimeout,
                        MaxRetries,
                        SNAPHeaderPresent,
                        UserPriority,
                        ACKRequested,
                        ConfirmRequested,
                        Length,
                        Data
                        )
```

The primitive parameters are defined in Figure 64.

6.3.2 MESH-ASYNC-DATA.confirm

This primitive is used to report the result of a request to transfer a MHSDU from one mesh entity to another mesh entity or entities. This primitive is only generated if the ConfirmRequested parameter in the MESH-ASYNC-DATA.request with the same RequestID value is ALWAYS or is ON_ERROR and the ResultCode is FAILURE. The semantics of this primitive are:

```
MESH-ASYNC-DATA.confirm (
                        RequestID,
                        TransmitDelay,
                        ResultCode,
                        ReasonCode
                        )
```

The primitive parameters are defined in Figure 64.

6.3.3 MESH-ASYNC-DATA.indication

This primitive is used to indicate the reception of an asynchronous MHSDU. The semantics of this primitive are:

```
MESH-ASYNC-DATA.indication (
                           MeshID,
                           TrgtTREEID,
                           TrgtID,
                           OrigTREEID,
                           OrigID,
                           SNAPHeaderPresent,
                           Length,
                           Data
                           )
```

The primitive parameters are defined in Figure 64.

6.3.4 MESH-ISOCH-DATA.request

This primitive is used to initiate the transfer of an isochronous MHSDU from one mesh entity to another mesh entity. The semantics of this primitive are:

```
MESH-ISOCH-DATA.request (
                        MeshID,
                        RequestID,
                        StreamIndex,
                        TransmitTimeout,
                        MaxRetries,
                        SNAPHeaderPresent,
                        ACKRequested,
                        ConfirmRequested,
                        Length,
                        Data
                        )
```

The primitive parameters are defined in Figure 64.

6.3.5 MESH-ISOCH-DATA.confirm

This primitive is used to report the result of a request to transfer an isochronous MHSDU from one mesh entity to another mesh entity or entities. This primitive is only generated if the ConfirmRequested parameter in the MESH-ISOCH-DATA.request with the same RequestID value is ALWAYS or is ON_ERROR and the ResultCode is FAILURE. The semantics of this primitive are:

```
MESH-ISOCH-DATA.confirm (
                        RequestID,
                        StreamIndex,
                        TransmitDelay,
                        ResultCode,
                        ReasonCode
                        )
```

The primitive parameters are defined in Figure 64.

6.3.6 MESH-ISOCH-DATA.indication

This primitive is used to indicate the reception of an isochronous MHSDU. The semantics of this primitive are:

```
MESH-ISOCH-DATA.indication (
                           MeshID,
                           TrgtTREEID,
                           TrgtID,
                           OrigTREEID,
                           OrigID,
                           StreamIndex,
                           SNAPHeaderPresent,
                           Length,
                           Data
                           )
```

The primitive parameters are defined in Figure 64.

6.4 Mesh PIB

The mesh PIB comprises the managed objects, attributes, actions, and notifications required to manage the mesh sublayer of a mesh capable DEV or mesh capable PNC. In the Access column of the Table 65, read only indicates that the parameter is only allowed to be read by the DME while read/write indicates that the DME is able to change parameters using the MHME-SET.request primitive.

Table 65—Mesh PIB parameters

Managed object	Octets	Definition	Access
MESHPIB_MeshID	2	Specifies the identification of a mesh network, which is unique value in the mesh netwrok	Read only
MESHPIB_TREEID	2	Specifies the unique identification of an MPNC on the tree topology	Read only
MESHPIB_MaxChild	1	Indicates the maximum number of child MDEV/MPNC	Read only
MESHPIB_MaxTreeLevel	1	Indicates the maximum number of hops that an MC can have	Read only
MESHPIB_MeshPNCCapable	1 bit	If the bit set to one, the PNC is mesh capable	Read only
MESHPIB_MeshDEVCapable	1 bit	If the bit set to one, the DEV is mesh capable	Read only
MESHPIB_MeshNeighborList	Set	Lists a TREEID block, tree level, link quality, and relationship for each neighbor MPNC	Read only

6.5 Frame formats

6.5.1 General mesh frame format

The mesh frames are described as a sequence of fields in a specific order. Each figure depicts the fields as they appear in the MAC frame and in the order in which they are transmitted in the wireless medium, from right to left where the right-most bit is transmitted first in time, as described in 7.1 of IEEE Std 802.15.3-2003.

The general MAC frame defined in IEEE Std 802.15.3-2003 is extended with a new mesh header field inserted at the beginning of the MAC frame payload. The format of the MAC frame body is shown in Figure 64.

octets: L_n	12
Mesh frame payload	Mesh header

Figure 64—Mesh frame format

The Mesh Header should be formatted as illustrated in Figure 65.

octets: 1	2	1	1	2	2	2	1
TTL	Mesh sequence number	Destination ID	Source ID	Destination TREEID	Source TREEID	MeshID	Mesh frame control

Figure 65— Mesh header format

The Mesh Frame Control field should be formatted as illustrated in Figure 66.

Bits: b7–b5	b4–b3	b2–b0
Reserved	Frame type	Transmission method

Figure 66— Mesh frame control field format

The Transmission Method subfield indicates how to transmit the frame. The valid values of Transmission Method subfield are defined in Table 66.

Table 66—Valid values of transmission method

Value $b_2\ b_1\ b_0$	Transmission method description
000	Full broadcast
001	Tree broadcast
010	Tree routing
011	Server routing
100 – 111	Reserved

The Frame Type subfield indicates the type of frame that is being sent. Table 103 lists the valid values of Frame Type subfield and their description.

Table 67 — Valid values of frame type

Type value b_4 b_3	Frame type description	Subclause
00	command	6.5.2.1
01	data	6.5.2.2
10	LLC/SNAP data	6.5.2.3
11	Reserved	

The MeshID field indicates the ID of the mesh network.

The Source TREEID field indicates the TREEID of the source MPNC.

The Destination TREEID field indicates the TREEID of the destination MPNC.

The Source ID field indicates the DEVID of the source MDEV.

The Destination ID field indicates the DEVID of the destination MDEV.

The Mesh Sequence Number field specifies a unique identifier that is used to prevent redundant frame relays and also enable sequential frame delivery.

The TTL field contains the maximum number of hops allowed during the transmission of this frame, as described in 6.6.6.1.

6.5.2 Format of individual frame types

6.5.2.1 Command format

The Mesh command format is shown in Figure 67.

octets: L	1	1	12
Mesh command payload	Length	Command type	Mesh header

Figure 67—Mesh command format

6.5.2.2 Data frame format

The Mesh Data frame uses the format in Figure 64 with the Mesh Frame Payload containing the mesh data.

6.5.2.3 LLC/SNAP data frame format

The Mesh LLC/SNAP Data frame is identical to a Mesh Data frame, as defined in 6.5.2.2, except that the Data Payload field includes an LLC/SNAP header as the first octets in the payload.

6.5.3 Command types

The valid values of command type are shown in Table 68.

Table 68—Command types

Command type	Command name	Subclause
0x00	TREEID request	6.5.3.1
0x01	TREEID assignment	6.5.3.2
0x02	Server notification	6.5.3.3
0x03	Server inquiry	6.5.3.4
0x04	Link state request	6.5.3.5
0x05	Link state registration	6.5.3.6
0x06	Route discovery	6.5.3.7
0x07	Route notification	6.5.3.8
0x08	Route formation	6.5.3.9
0x09	Route error	6.5.3.10
0x0A-0xFF	Reserved	–

6.5.3.1 TREEID request command

The TREEID Request command is used by a child MPNC to request a TREEID block from its parent MPNC in a tree network. The TREEID Request command should be formatted as illustrated in Table 68.

octets: 2	1	1
Descendant number	Length	Command type

Figure 68—TREEID request command format

The Descendant Number field indicates the actual or expected number of descendant MPNCs. It may be set as zero if the MPNC is not clear about how many descendants it will have.

6.5.3.2 TREEID assignment command

The TREEID Assignment command is used by a parent MPNC to assign a TREEID block to one of its child MPNCs. The TREEID Assignment command should be formatted as illustrated in Figure 69.

octets: 2	2	1	1
Ending TREEID	Starting TREEID	Length (= 6)	Command type

Figure 69—TREEID assignment command format

The Starting TREEID field indicates the lowest TREEID of the assigned TREEID block, as described in 6.6.3.

The Ending TREEID field indicates the highest TREEID of the assigned TREEID block, as described in 6.6.3.

6.5.3.3 Server notification command

The Server Notification command is used to carry the server information to one or more MPNCs. The Server notification command should be formatted as illustrated in Figure 70.

Octets: 2	1	1	1
Server TREEID	Server type	Length	Command type

Figure 70—Server notification format

The Server Type field indicates the type of server. The valid values of Server Type field are shown in Table 69.

The Server TREEID field indicates the identity of the server.

Table 69—Valid values of server type

Type value	Server type description
0x00	MC
0x01	Topology server
0x02	Gateway
0x03 – 0xFF	Reserved

6.5.3.4 Server inquiry command

The Server Inquiry command is used by a child MPNC to ask for server information from its parent MPNC. The Server Inquiry command should be formatted as illustrated in Figure 71.

octets: 1	1	1
Server type	Length	Command type

Figure 71—Server inquiry command format

The Server Type field is defined in 6.5.3.3.

6.5.3.5 Link state request command

The Link State Request command is used by a topology server to request link state information from the descendant MPNCs located on the subtree rooted at itself.

octets: 1	1	1
Cost type	Length	Command type

Figure 72—Link state request command format

The Cost Type field indicates the type of link cost. The valid values of Cost Type field are shown in Table 70.

Table 70—Valid values of link cost type

Type value	Link cost	Description
0x00	Hop count	Number of links between two MPNCs
0x01	LQI	Quality of the received signal as defined in 6.7 (Table 38) of IEEE Std 802.15.3-2003
0x02	Residual CTA time	CTA time in milliseconds available at MPNC
0x03	Delay	Delay in number of superframes as defined in 6.6 (Table 36) of IEEE Std 802.15.3b-2005
0x04 – 0xFF	Reserved	–

6.5.3.6 Link state registration command

The Link State Registration command is used by an MPNC to register its link state information at the topology server. The Link State Registration command should be formatted as illustrated in Figure 73.

octets: 3*n	1	2	2	1	1
Link status list	Link cost type	Terminator TREEID	Initiator TREEID	Length	Command type

Figure 73—Link state registration command format

The Initiator TREEID field identifies the MPNC that sends the link state registration command.

The Terminator TREEID field identifies the MPNC that stops the link state registration process. Terminators are usually the topology servers that initiate link state registration process.

The Cost Type field indicates the type of link cost, as defined in Table 70.

The format of the Link Status List field is shown in Figure 74.

octets: 2	1	…	1	2	1	2
Neighbor link cost n	Neighbor TREEID n	…	Neighbor link cost 2	Neighbor TREEID 2	Neighbor link cost 1	Neighbor TREEID 1

Figure 74—Link status list subfield format

The Neighbor TREEID subfield identifies the identity of its neighbor MPNC.

The Neighbor Link Cost subfield identifies the link cost between itself and its neighbor MPNC.

6.5.3.7 Route discovery command

The Route Discovery command is used to create routing entries for a source MPNC, and/or discover an optimal route to a destination MPNC. The Route Discovery command should be formatted as illustrated in Figure 75.

octets: 1	2	2	2	1	1
Cost type	Route destination TREEID	Route source TREEID	Sequence Number	Length	Command type

Figure 75—Route discovery command format

The Sequence Number field specifies a unique sequence identifier that is used to match the Route Notification command or the Route Formation command.

The Route Source TREEID field identifies the source MPNC of the desired route.

The Route Destination TREEID field identifies the destination MPNC of the desired route.

The Cost Type field indicates the type of route cost. The valid values of Cost Type field are shown in Table 71.

Table 71—Valid values of route cost type

Type value	Route cost	Description
0x00	Hop count	Sum of link costs on the route between source and destination MPNCs
0x01	LQI	Minimum of link costs on the route between source and destination MPNCs
0x02	Residual CTA time	Minimum of link costs on the route between source and destination MPNCs
0x03	Delay	Sum of link costs on the route between source and destination MPNCs
0x04 – 0xFF	Reserved	–

6.5.3.8 Route notification command

The Route Notification command is used by a topology server to notify the destination MPNC of the optimal route between the source and destination MPNCs. The Route notification command should be formatted as illustrated in Figure 76.

octets: 4*n	2	2	2	1	1	1
Relay list	Route cost	Route cost type	Route source TREEID	Sequence number	Length	Command type

Figure 76—Route notification command format

The Sequence Number field is copied from the corresponding Route Discovery command.

The Route Source TREEID field identifies the source MPNC of the optimal route.

The Cost Ttype field indicates the type of route cost. The valid values of cost type subfield are shown in Table 71.

The Route Cost field records the cost of the whole route.

The Relay List field indicates the optimal route between the source and destination MPNCs. The Relay List field should be formatted as illustrated in Figure 77.

octets: 2	**2**	**…**	**2**	**2**	**2**	**2**
Relay route cost n	Relay TREEID n	…	Relay route cost 2	Relay TREEID 2	Relay route cost 1	Relay TREEID 1

Figure 77—Relay list field format

The Relay TREEID subfield indicates the identity of each relay.

The Relay Route Cost subfield indicates the cost from each relay to the destination MPNC, as defined in Table 107.

6.5.3.9 Route formation command

The Route Formation command is used to deliver route information to relays along a newly discovered route, so that these relays may establish routing entries accordingly. The Route Formation command should be formatted as illustrated in Figure 78.

Octets: 4*n	**2**	**1**	**2**	**2**	**2**	**1**	**1**
Relay list	Route cost	Route cost type	Route destination TREEID	Route source TREEID	Sequence number	Length	Command type

Figure 78—Route formation command format

The Sequence Number field is copied from the corresponding Route Discovery or Route Notification command s.

The Route Source TREEID field identifies the source MPNC of the desired route, which terminates the frame.

The Route Destination TREEID field identifies the destination MPNC of the desired route, which originates the frame.

The Cost Type field indicates the type of route cost. The valid values of Cost Type field are shown in Figure 69.

The Route Cost field records the route cost from the frame transmitter to the source MPNC and is defined in 6.5.3.8.

The Relay List field contains the address of each relay device, as well as its route cost to the destination and is defined in 6.5.3.8.

6.5.3.10 Route error command

The Route Error command is used by a relay MPNC to notify the source MPNC about route errors. The Route Error command should be formatted as illustrated in Figure 79.

octets: 1	2	2	1	1
Error reason	Terminator TREEID	Initiator TREEID	Length	Command type

Figure 79—Route error command format

The Initiator TREEID field identifies the MPNC that detects and reports the failed link.

The Terminator TREEID field identifies the MPNC located at the other end of the failed link.

The valid values of the Error Reason field are shown in Table 72.

Table 72—Valid values for error reason

Value	Error reason description
0x00	Link break
0x01	Low capacity
0x02	Low power
0x03 – 0xFF	Reserved

6.6 Mesh service support

This subclause describes the functions the mesh sublayer provides to the DME. These functions include network initialization/termination, topology formation, unicast addressing and routing algorithms.

6.6.1 Starting a mesh network

A mesh network usually starts by a device that is capable of operating as an MC. The DME of the device that is ready to start its own mesh network should issue a MHME-SCAN.request, as described in 6.2.3.1, to determine the presence or absence of other mesh networks. The mesh sublayer will then request the MAC sublayer to scan by issuing the MLME-SCAN.request primitive to the MAC sublayer.

When the MLME-SCAN.confirm primitive comes back from the MAC sublayer, the mesh sublayer will gather information about the mesh networks operating in its neighborhood. The useful information includes mesh ID, TREEID block, PNID, and the channel each mesh network is operating on, etc. The mesh sublayer notifies the DME of the mesh-related parameters by issuing the MHME-SCAN.confirm primitive to the DME.

If there is at least one mesh network found from the scan procedure, the DEV may join one of them. Otherwise, the DEV determines to opearate as an MC. The MC begins the mesh initialization procedure in which the parameters needed to operate a mesh network are determined. Such parameters include the mesh ID, TREEIDs, and the opearating channel, etc.

The DME can now start the mesh network by issuing the MHME-START.request primitive, as described in 6.2.4.1, to the mesh sublayer. On the receipt of this primitive, the mesh sublayer issues the MLME-MESH-CAPABILITY.request primitive to the MAC sublayer with the mesh-related parameters such as a mesh ID and a TREEID block included. Upon receiving this primitive, the MAC sublayer enters the operating mode and initializes the parameters to operate itself as an MPNC.

After successfully passing the mesh parameters for MPNC initialization to the MAC sublayer, the mesh sublayer issues the MLME-START.request primitive to the MAC sublayer. The MAC sublayer then attemps to start a piconet as described in 8.2.2 of IEEE Std 802.1.5.3-2003. Upon the receipt of the MLME-START.confirm primitive from the MAC sublayer indicating the results of the beacon generation procedure, the mesh sublayer passes this information to the DME by issuing the MHME-START.confirm.

Figure 80 illustrates a sequence of message flow for starting a mesh network by the MC.

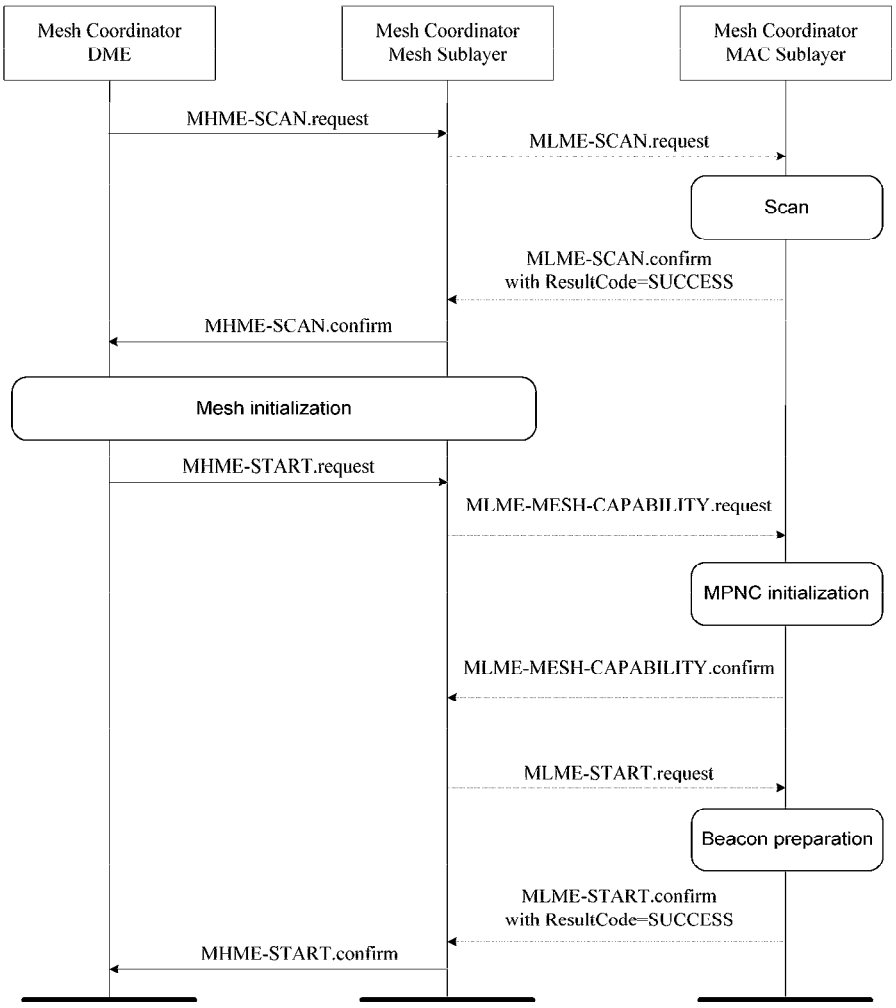

Figure 80—Message sequence chart for starting a mesh network

6.6.2 Tree formation

A mesh network is constructed on the basis of tree topology with the MC as its root. Forming a tree-based mesh network usually consists of the following steps:

— Searching for existing channels/piconets

— Selecting a parent piconet to join

— Association with the parent MPNC

— TREEID assignment

— Creating a new child piconet

The first thing a non-MC device should do is to detect whether there are WPAN networks operating in its vicinity. The DME of the device will issue a MHME-SCAN.request to its mesh sublayer. The mesh sublayer will attempt to discover networks operating within the device's operating area by performing the passive scan over the channels specified in the MHME-SCAN.request primitive. This scan is performed by issuing the MLME-SCAN.request primitive to the MAC sublayer of the device.

When the MLME-SCAN.confirm primitive comes back from the MAC sublayer, the mesh sublayer will gather information about the mesh networks operating in its neighborhood. The useful information includes mesh ID, TREEID block, PNID, and the channel each mesh network is operating on, etc. The mesh sublayer notifies the DME of the mesh-related parameters by issueing the MHME-SCAN.confirm primitive to the DME.

After completing the scan for mesh networks, the device associates with one of the MPNCs detected through the scan procedure, request a block of TREEIDs, and create a new child piconet if the device is going to operate as the MPNC.

Based on the information given in the MHME-SCAN.confirm, the DME will try to find the best MPNC with which to associate, if multiple candidates are available. The criteria for selecting the parent MPNC may include the MPNC's level on the tree, link quality between the device and the MPNC, and the residual CTA time at the MPNC, etc. The implementers may take other factors into account if their application imposes special requirements. The detailed procedure for the parent MPNC selection is beyond the scope of this document. Association is allowed only if the tree level of the chosen MPNC is less than the MESHPIB_MaxTreeLevel, as defined in 6.4.

After determining the parent MPNC with which to associate, the DME of the device issues an MHME-ASSOCIATE.request primitive, as described in 6.2.6.1, to its mesh sublayer with the parameters properly set. Then the mesh sublayer issues the MLME-ASSOCIATE.request primitive to the MAC sublayer to start the association procedure at the MAC sublayer.

After successfully associating as a member of a piconet, the DME of the device issues an MHME-TREEID-ASSIGN.request primitive, as described in 0, to the mesh sublayer. Then the mesh sublayer starts a TREEID assignment procedure by sending a TREEID Request command to the parent MPNC. If the parent MPNC has sufficient TREEIDs available in its TREEID block, it responds with a TREEID Assignment command specifying the TREEID block to be assigned for the requesting device. Upon the receipt of the TREEID Assignment command, the mesh sublayer passes this TREEID information to the DME by issueing the MHME-TREEID-ASSIGN.confirm primitive to the DME. The detailed procedure of TREEID assignment is described in the 6.6.3.

Upon receiving the MHME-TREEID-ASSIGN.confirm primitive, the DME of the device creates a child piconet, if allowed. The DME starts the creation procedure of child piconet by issuing the MHME-START.request primitive to the mesh sublayer. On the receipt of this primitive, the mesh sublayer issues

the MLME-MESH-CAPABILITY.request primitive to the MAC sublayer with the mesh-related parameters such as a mesh ID and a TREEID block included. Upon receiving this primitive, the MAC sublayer enters the operating mode and initializes the parameters to operate as a child MPNC.

After successfully passing the mesh parameters for MPNC initialization to the MAC sublayer, the mesh sublayer issues the MLME-START.request primitive to the MAC sublayer. Upon receiving the MLME-START.request primitive, the MAC sublayer prepares itself for transmitting beacons. Then the MAC sublayer returns the MLME-START.confirm primitive to the mesh sublayer indicating the results of a beacon generation procedure. The mesh sublayer passes this information to the DME by issuing the MHME-START.confirm.

Figure 81 illustrates a sequence of message flow to form a tree topology in a mesh network.

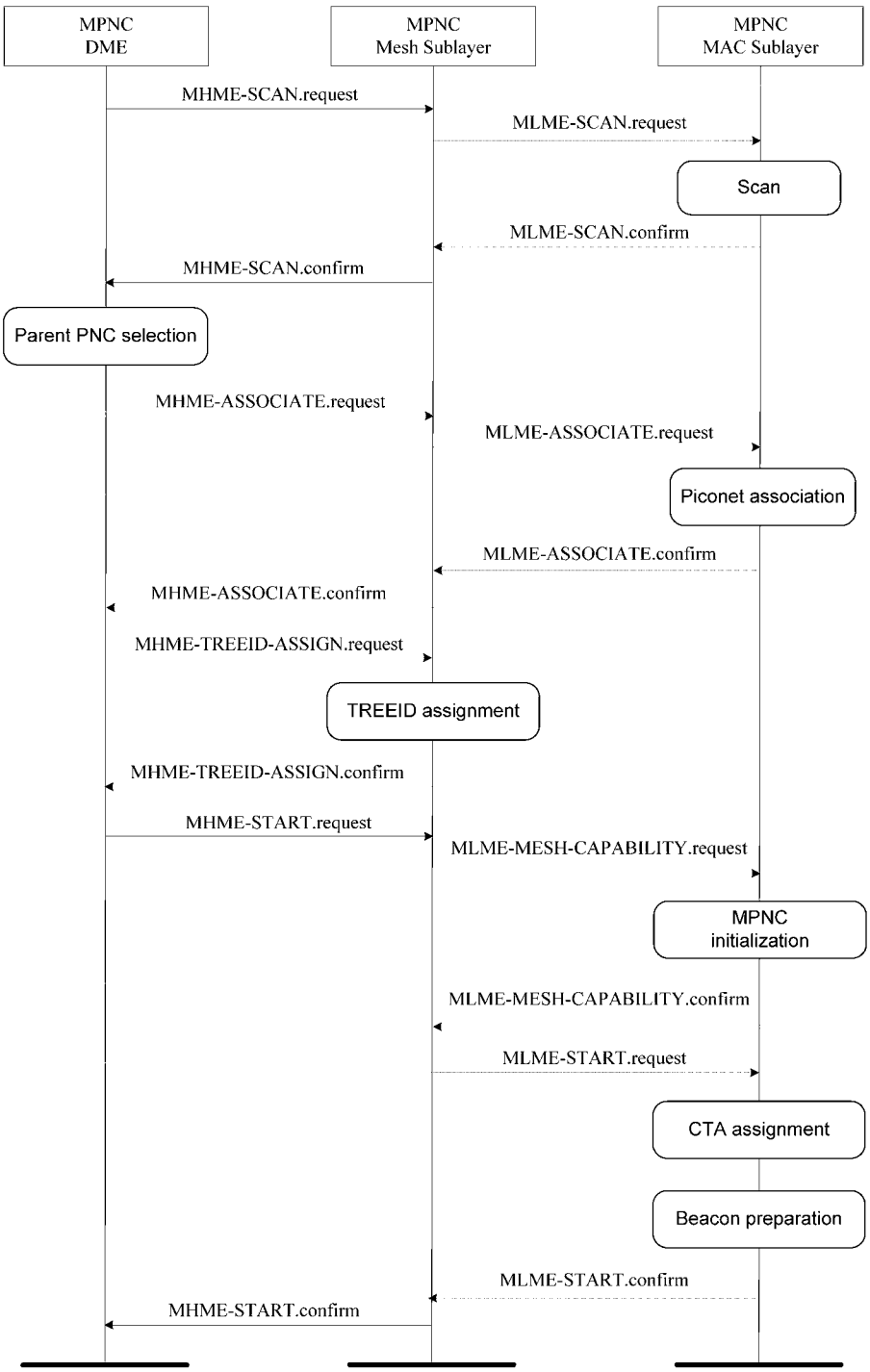

Figure 81—Message sequence chart for tree formation

6.6.3 TREEID assignment

An MPNC requires a TREEID to uniquely identify itself in the mesh network and also enable the tree-based self-routing. Every MPNC maintains at the mesh sublayer a block of TREEIDs of which the lowest one should be reserved for itself. An MC generates its own TREEID block whose size is upto 65536

(0~65535). When constructing a tree topology, other MPNCs should be assigned a unique TREEID block from its parent MPNC in a top-down manner starting from the MC.

When a device associates with the MPNC and intends to operate as a child MPNC, it sends the TREEID Request command to the parent MPNC with the Descendant Number field set to a proper value which is less than or equal to the MESHPIB_MaxChild, as defined in 6.4. The Descendant Number field is usally composed of two parts:

$$P_{dnum} = C_{dnum} + R_{dnum}$$

P_{dnum} is the derived descendant number of the child MPNC. C_{dnum} is the required number of TREEIDs expected by the child MPNC. R_{dnum} is the number of TREEIDs the child MPNC wants to reserve for future descendants.

When the parent MPNC receives a TREEID Request command , it checks the availability of the TREEIDs in its own TREEID block. If there are not enough TREEIDs available, the parent MPNC may reject to assign TREEIDs and respond with a TREEID Assignment command with both Starting TREEID and Ending TREEID fields set to NULL. The parent MPNC may also send the TREEID Request command to its parent MPNCs to obtain a larger TREEID block. Otherwise, the parent MPNC assigns a portion of available TREEIDs from its TREEID block and responds with a TREEID Assignment command that contains the allocated TREEID block. Figure 82 indicates the directions of the command s related to the TREEID assignment procedure.

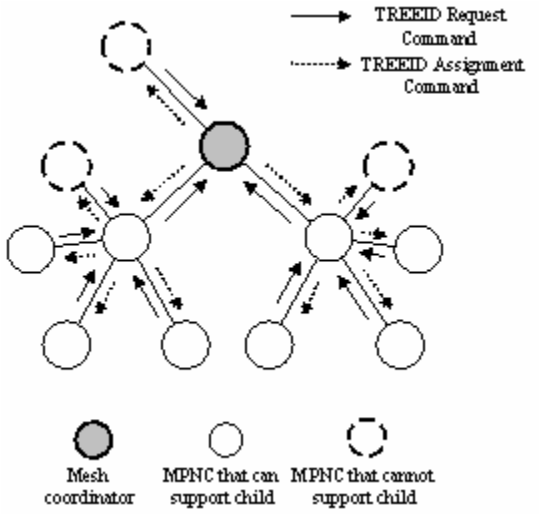

Figure 82—TREEID assignment procedure

Specifically, the parent MPNC assigns single TREEIDs to those children who cannot support any child; and then, it reserves R_{dnum} TREEIDs for future descendants; lastly, it divides the remaining TREEIDs among the children which have descendants or may support descendants. The parent MPNC may divide the TREEIDs based on the descendant ratios of these children. Each of these children is assigned a TREEID block which contains its own TREEID and the TREEIDs of its existing and potential descendants. The parent MPNC should record the TREEID assignment in its neighbor list, as defined in 6.6.5, and send a TREEID Assignment command to each child to notify the assigned TREEID block.

Once the child MPNC receives a TREEID block sucessfully, it sets aside a portion of the assigned TREEID block for its descendants so that the child MPNC assigns TREEIDs to newly joined children. The child MPNC should record its parent TREEID, as well as its children's TREEID blocks in the neighbor list, too. It should also send a TREEID Assignment command to each child to notify the assigned TREEID block. Eventually, every MPNC in the tree obtains its TREEID block from its parent MPNC, respectively.

Figure 83 shows an example of TREEID assignment in a tree topology network. In this example, the MC creates a TREEID block ranging from 0 to 37 with TREEID 0 reserved for itself. It first assigns TREEID 1 to the device that cannot support any children. Next, the MC assigns one device the TREEID block of [2, 16] and the other the TREEID block of [17, 31]. The MC reserves a TREEID block of [32, 37] for future use. The device with TREEID 2 assigns again its children three sets of TREEID blocks ([4, 6], [7, 9], [10, 12]) selecting from its TREEID block. It also reserves a TREEID block of [13, 16] for future use.

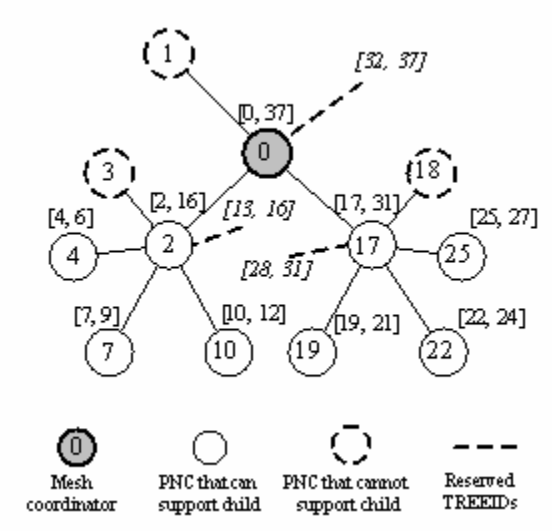

Figure 83—TREEID assignment example

As TREEIDs always conform to the tree hierarchy, they may change with variations of the tree structure. To cope with the topology variation, any MPNCs in the tree are also allowed to initiate the TREEID assignment process, if necessary, to adjust the TREEIDs that are assigned to its children under the sub-tree.

The dynamics of wireless channels and mobility of devices may cause tree structure changes. If an MPNC cannot get contact with its parent after mMaxLostBeacons superframes, it may select a new parent from its neighbor MPNCs and try to associate with the new parent. If the new parent reserves a large TREEID block which may accommodate the MPNC and all its descendants, the MPNC may obtain its new TREEID block from the new parent immediately. The MPNC should divide the new TREEID block among its children, and its children do the same thing among their own children. Eventually, all descendants of the MPNC obtain their new TREEID blocks. If the neighbors of the MPNC cannot support the MPNC and its descendants, the MPNC may disassociate its children and let the children find their new parents. In this case, both devices should delete the record of each other from their neighbor lists. An MPNC may maintain a list of backup parents to accelerate the process of tree reconstruction.

6.6.4 Leaving a mesh network

An MPNC may leave the mesh network due to a shutdown request by the DME. This subclause specifies two cases of disassociation: the shutdown initiated by the MPNC itself and the disassociation forced by the parent MPNC.

6.6.4.1 MPNC Shutdown

The DME of an MPNC starts the shutdown process by issueing a MHME-STOP.request primitive, as described in 6.2.5.1, to the mesh sublayer. The mesh sublayer in turn issues the MLME-STOP.request primitive to the MAC sublayer with the RequestType pramter set to SHUTDOWN. Upon the receipt of the MLME-STOP.request primitive, the MAC sublayer follows the shutdown process described in 8.2.7 of IEEE Std 802.15.3-2003. In addition, if the shutdown MPNC is not the MC, it should terminate the CTA allocated from the parent MPNC by sending a Channel Time Request command to the parent MPNC.

Figure 84 illustrates a sequence of message flow for an MPNC to stop its own piconet.

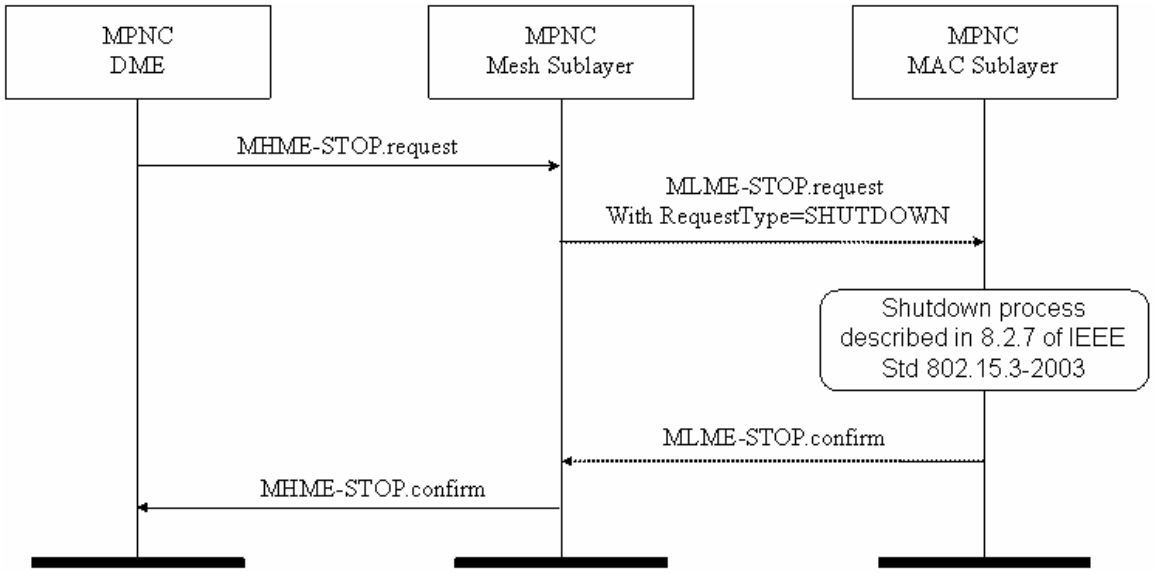

Figure 84—Message sequence chart for MPNC shutdown (parent side)

When a child MPNC receives the beacon with PNC Shutdown IE from a parent MPNC, the MAC sublayer of the child MPNC issues the MLME-DISASSOCIATE.indication primitive including the DEVID and DEVAddress of the parent MPNC. The mesh sublayer passes this information to the DME via MHME-DISASSOCIATE.indication primitive. Once the DME of the child MPNC receives this primitive, it determines whether the parent MPNC is the MC or not.

If the parent MPNC is the MC, the child MPNC with the lowest TREEID should take over as the new MC of the mesh network. After initialization mesh-related parameters, the new MC starts to transmit beacons to the neighboring MPNCs. The rest of child MPNCs that were not chosen as the new MC should wait for the beacons generated by the new MC. Once the child MPNC that lost its parent MPNC receives the beacons from the new MC, it should associate with the new MC and request a new TREEID block. If the child MPNC is assigned a TREEID block from the new MC, it should reassign the TREEIDs to its child MPNCs by sending the TREEID Assignment commands. This process of TREEID assignment is repeated until the TREEID Assignment command reaches the MPNC in the leaf.

If the parent MPNC is not the MC, the child MPNC immediately selects a new parent MPNC, if exists, among the neighboring MPNCs. The selection criteria for the new parent MPNC is beyond the scope of this document. After associating with the new parent MPNC, it should also request a new TREEID block. If the child MPNC is assigned a TREEID block from the new parent MPNC, it should reassign the TREEIDs to its child MPNCs by sending the TREEID Assignment command. This process of TREEID assignment is repeated until the TREEID Assignment command reaches the MPNC in the leaf of the tree.

Figure 85 illustrates a sequence of message flow for the child MPNC to handle the shutdown request from the parent MPNC.

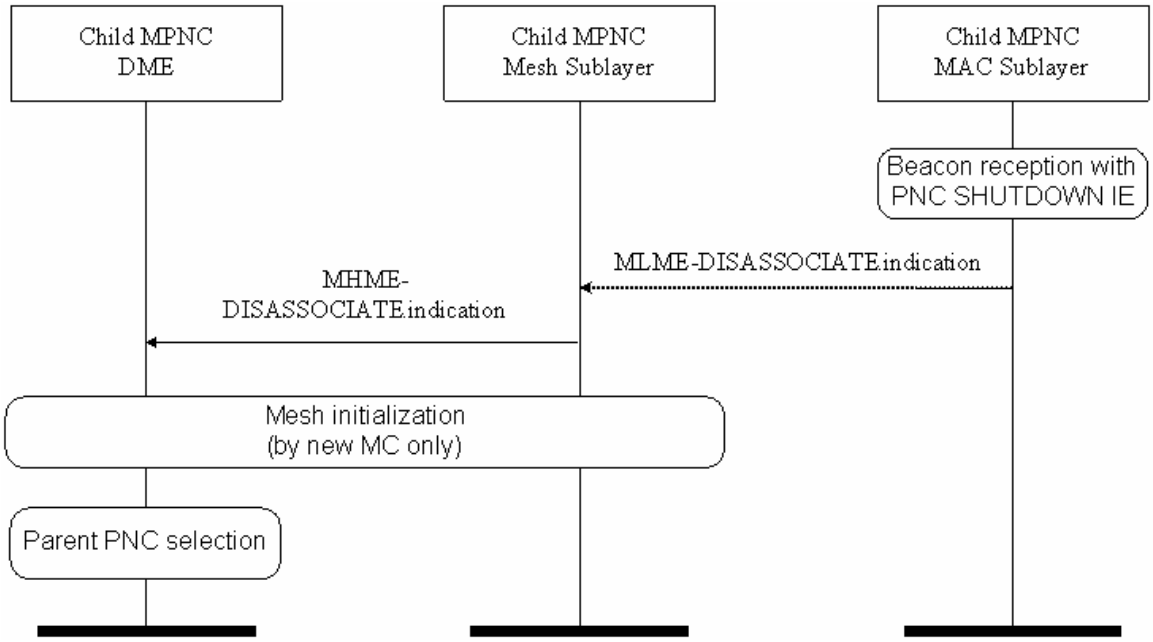

Figure 85—Message sequence chart for MPNC shutdown (child side)

6.6.4.2 Parent-initiated disassociation

When an MPNC wants to remove one of child MPNCs from the mesh network, it uses the process described in 8.2.7.2 of IEEE Std 802.15.3-2003.

6.6.5 Routing procedure

When a source MPNC wants to deliver a packet by using tree routing, it should set the Transmission Method in the Mesh Header field to indicate tree routing, as defined in 6.5.1.

The tree structure and TREEID assignment method enable MPNCs to route packets by the comparison of TREEID blocks. To conduct tree routing, an MPNC first checks whether the Destination TREEID field in the mesh header of the incoming frame falls into one of the following TREEID blocks: downtree, neighbor, or uptree. If the destination TREEID falls into one of its children's TREEID blocks, the MPNC should forward the packet to the corresponding child MPNC. Otherwise, the MPNC checks whether the destination TREEID falls into one of its neighbor's TREEID blocks. If so, the MPNC should forward the packet to the corresponding neighbor MPNC. Otherwise, that is, if the destination TREEID belongs to none of its children's TREEID blocks and none of its neighbor's TREEID blocks, the MPNC should forward the packet to the parent MPNC.

For an MPNC to forward mesh frames to the neighbor MPNC that belongs to a different parent from itself, the sending MPNC may have additional parents for optimal routing. In addition to the current parent MPNC on the tree, the sending MPNC is allowed to associate temporarily with the receiving MPNC, as illustrated in Figure 86. In this case, however, the sending MPNC is restricted to opearate only as an MDEV of the receiving MPNC. Then the sending MPNC requests the receiving MPNC to assign the CTA. After forwarding the frames using the assigned CTA, the sending MPNC disassocciates from the receiving MPNC.

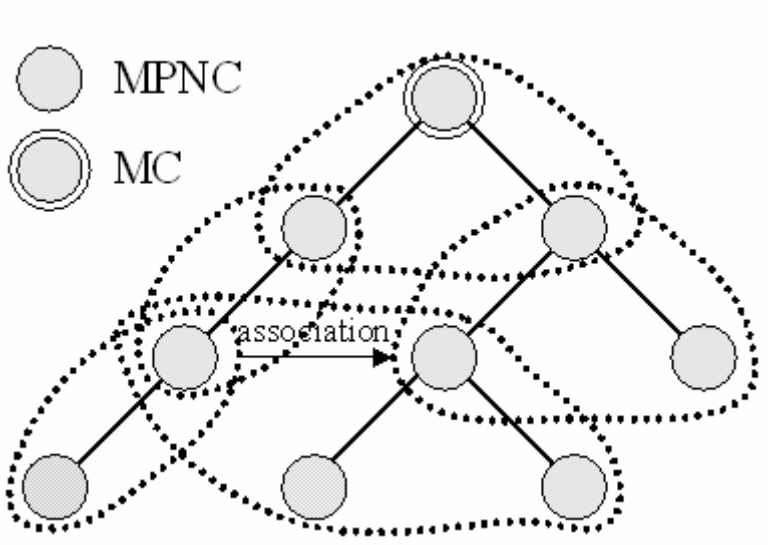

Figure 86—Association for optimal routing

Figure 87 shows the routing process for the packet to be forwarded.

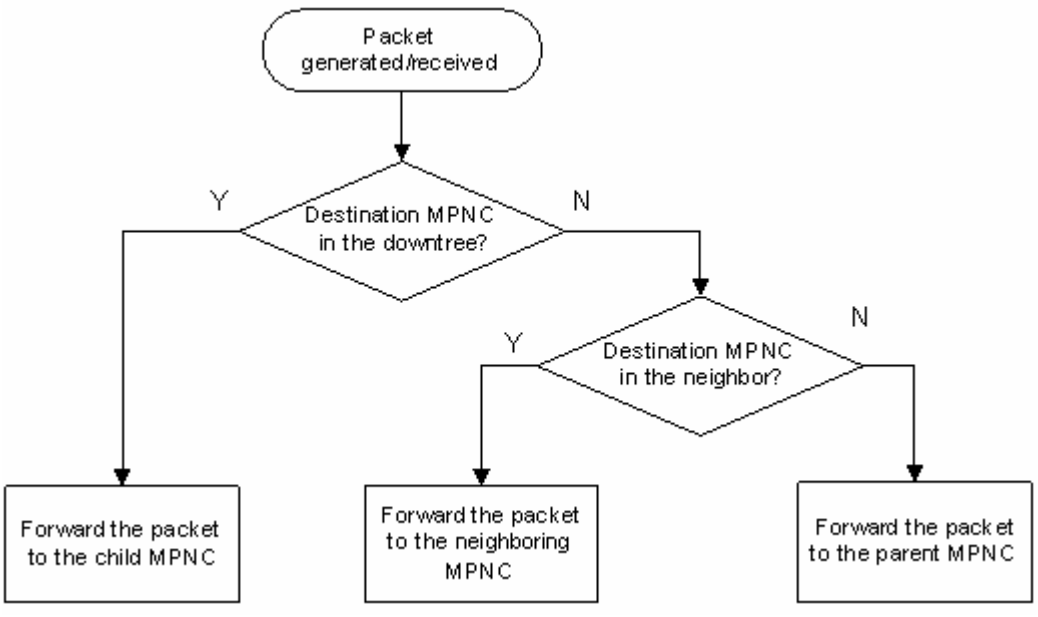

Figure 87—Packet forwarding procedure

To compare the TREEID of the destination MPNC with those of neighbor MPNCs, each MPNC should maintain the neighbor list as shown in Table 73. Each MPNC should update its neighbor list upon the reception of beacon frames from the neighbor MPNCs. The Starting TREEID and Ending TREEID fields indicate the TREEID block maintained by each neighbor MPNC. The Tree Level field indicates the depth of tree at which the neighbor MPNC is located. The Link Quality field indicates the link quality from a specific neighbor to this node. It is measured when the data frames are received from neighbors and quantified into 16 levels, from 0 (the lowest) to 15 (the highest). The Relationship field indicates the relationship between this node and a specific neighboring MPNC.

Table 73—Neighbor list

Starting Address	Ending Address	Tree Level	Link Quality	Relationship
Starting $TREEID_1$	Ending $TREEID_1$	Hop count to MC_1	LQ_1	parent/child/neighbor
Starting $TREEID_2$	Ending $TREEID_2$	Hop count to MC_2	LQ_2	parent/child/neighbor
…	…	…	…	…
Starting $TREEID_n$	Ending $TREEID_n$	Hop count to MC_n	LQ_n	parent/child/neighbor

Figure 88 shows an example of tree routing. In this example, MPNC 7 wants to send a packet to MPNC 19. MPNC 7 checks its TREEID block and finds that MPNC 19 does not fall into its downtree TREEID block. So MPNC 7 forwards the packet to its parent, MPNC 2. MPNC 2 also finds that MPNC 19 does not fall into its downtree TREEID block. So MPNC 2 forwards the packet to its parent, MPNC 0. MPNC 0 finds that MPNC 19 falls into the downtree TREEID block of MPNC 17, one of its children. So it forwards the packet to MPNC 17. Finally MPNC 17 forwards the packet to Device 19. In this case, however, if MPNC 7 is neighboring with MPNC 2, the packet would be forwarded directly from MPNC 2 to MPNC 7 without being relayed by the MPNC 0.

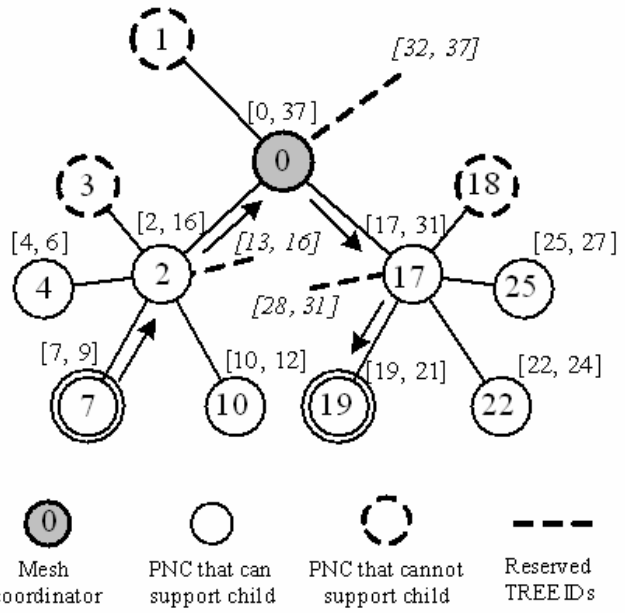

Figure 88—Tree routing example

6.6.6 Routing Alternative

The topology server is optionally introduced to provide the optimal route between two descendant MPNCs. With the existence of topology servers, MPNCs may consult these servers for routing information, instead of forwarding the packets on the tree-based route. Every MPNC in the tree nework can be selectively allowed to play the role of a topology server. This server routing consists of three processes: link state registration, route establishment, and route recovery.

6.6.6.1 Link state registration

An MPNC initiates the link state registration process by creating a Link State Request command and broadcasting the frame to its descendants. The Transmission Method field in the mesh header of this frame should be set as tree broadcast to ensure that only the children of the MC accept and process this frame. The destination TREEID and destination DEVID fields in the mesh header of the frame should be set as BcstID. The source TREEID and source DEVID subfields of the frame should be set as the identities of the MC.

When an MPNC in a tree receives a Link State Request command from its parent, the MPNC creates a Link State Registration command and sends the frame to its parent. The initiator TREEID of the frame should be set as the identity of the MPNC that initiated the link state registration process. Then, the MPNC should forward the Link State Request command to its children, if they exist.

The link state information contained in the Link State Registration command should be a certain type of link cost, as defined in 6.5.3.5. Whenever there is a big change in its link state thereafter, the MPNC should send a link state registration command to its ancestors to update its link state information.

Once the MPNC receives the link state registration command from its child, it should identify the initiator TREEID of the frame and store the accompanying link state information in the link state cost table which is not defined in this document. Then, the MPNC should forward the frame to its parent. This forwarding process is repeated until the Link State Registration command reaches to the MPNC that initiated the link state registration process. This MPNC is identified by the Terminator TREEID field in the mesh header of the Link State Registration command. To limit the amount of link state information maintained by each MPNC, the depth of subtree can be restricted to be a certain value. This is done by setting the TTL field in the mesh header of the Link State Request Command to a desired value..

Based on the link state cost table, an MPNC calculates the optimal route between any source-destination pair on the subtree consisting of its descendent MPNCs, by using the well-known shortest path computation method such as Bellman-Ford or Dijkstra algorithm. To limit the amount of link state information maintained by each MPNC, the depth of subtree may be set to a certain value. The computation results are stored in the routing table which is not defined in this document.

6.6.6.2 Route establishment

When a source MPNC wants to deliver a frame by using server routing, it should set the Transmission Method in the mesh header of the frame to indicate server routing, as defined in 6.5.1. A source MPNC can seek the help from one of its ancestors to locate the optimal route toward a desired destination. To provide the optimal route between any source-destination pair, the shortest path computation should be done by the common ancestor that is closest to both of them. Recall that an MPNC maintains link state information of its descendants only. Thus, the MPNC that provides the optimal route varies with the relative position of the source-destination pair on the tree. The MC is free from this location constraint since all MPNCs are its descendants.

A source MPNC should check the route destination of the frame against its routing table. If the MPNC has a routing entry for that destination, it calculates the optimal route between the source and destination MPNCs, and creates a Route Notification command to carry the calculation result. The MPNC should include a relay list in the Route Notification command to specify the identities of each relay, as well as the route cost from each relay to the destination MPNC. The MPNC should send the frame to the destination MPNC by using the tree structure.

If the source MPNC does not have a routing entry for the destination, it should send a Route Discovery command to its parent MPNC on the tree. The Route Source TREEID field of the frame should be set as the identity of the source MPNC. The Route Destination TREEID field of the frame should be set as the identity of the destination MPNC. The source MPNC may also indicate the preferred QoS metric to be used for the optimal route by setting the Cost Type field to a value, as defined in 6.5.3.7. The default QoS metric should be the hop count.

When the parent MPNC receives the Route Discovery command, it also checks the route destination of the frame against its routing table. If it does not have a routing entry for that destination, it should simply relay the frame to its parents. This process may be repeated until the frame reaches to the MC. Otherwise; it should create a Route Notification command that is sent to the destination MPNC using the tree structure.

When the destination MPNC receives the Route Notification command, it should check the route source TREEID against its routing table. If it already has a routing entry for the source MPNC, it should update the routing entry by consulting the relay list of the Route Notification command. If it does not have a routing entry for the source MPNC, it should create one. Then, the destination MPNC should create a Route Formation command, and send the frame to the source MPNC along the newly derived route. The destination MPNC should copy the relay list from the Route Notification command to the Route Formation command, and put the identity of the source MPNC and its own identity in the Route Source TREEID and Route Destination TREEID fields, respectively. The Route Formation command is delivered to the next-hop relay according to the newly created/updated routing entry for the source MPNC.

When an MPNC receives a Route Formation command, it should check the source and destination MPNCs against its routing table. If it already has routing entries for the source and destination MPNCs, it should update the routing entries by consulting the relay list of the route formation frame. If it does not have routing entries for the source and destination MPNCs, it should create new entries for them. The MPNC should forward the Route Formation command to the next-hop relay according to the newly created/updated routing entry for the source MPNC.

When the source MPNC receives the Route Formation command, it should create a routing entry for the destination MPNC by consulting the relay list of the route formation frame. After it creates the entry, it may begin to deliver data frames along the newly formed route.

Figure 89 and Figure 90 show two examples of the server routing. In Figure 89, the MC provides the optimal route since it is the only common ancestor for the source-destinaiton pair. However, if the source-destination pair belongs to a subtree as presented in Figure 90, the MPNC other than the MC may also operate as the topology server.

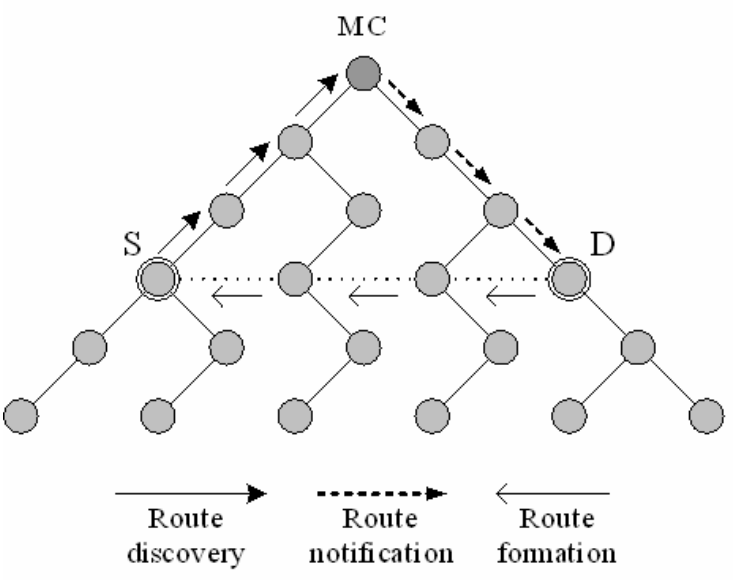

Figure 89—Server routing with the aid of MC

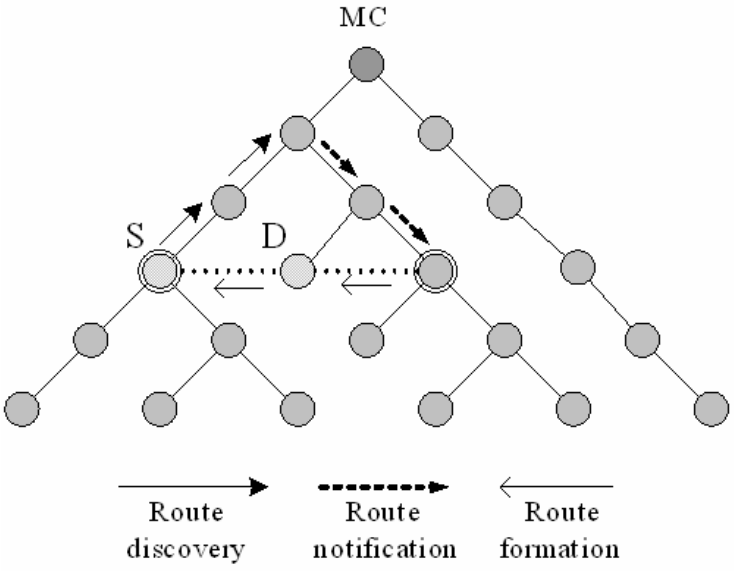

Figure 90—Server routing without the aid of MC

6.6.6.3 Route recovery

Whenever there is a failure in its link state, an MPNC should send a Route Error command to the topology server to update its routing information. The topology server recalculates the optimal routes for all source-destination pairs that are affected by the reported link state. Then it sends the Route Notification command to the destination MPNCs whose current optimal routes are different from the previous ones. The rest of route recovery process is the same as the route establishment process.

Annex A

(informative)

Amendment to MAC sublayer

This annex describes the recommended changes in IEEE Std 802.15.3-2003 MAC and IEEE Std 802.15.3b-2005 MAC required to support the mesh networking protocol described in this specification. The subclauses in this annex are in-line with the clauses in the IEEE Std 802.15.3-2003 MAC and IEEE Std 802.15.3b-2005 MAC standards to provide implementers easy cross-reference between these documents.

3. Definitions

Insert the following definitions in alphabetical order:

3.45 mesh network: An extended form of piconet that supports intra-piconet communication as well as inter-piconet communication between the neighboring mesh piconet coordinators.

Acronyms and abbreviations

Insert the following acronyms:

MDEV	mesh capable device
MPNC	mesh capable piconet coordinator

5. General description

Insert the following subclause 5.5 after the subclause 5.4:

5.5 Mesh networking capability extension

In addition to the MAC functionalities described in the subclause 5.3, the IEEE 802.15.3 MAC also allows for extension to support mesh network capabilities. A mesh network is an extended form of a piconet that supports intra-piconet communication as well as inter-piconet communication. The process of establishing and maintaining a mesh network is outside of the scope of this standard.

6. Layer management

6.3 MLME SAP interface

Insert the following low after the last low in Table 3 as shown. The other rows are unchanged and not shown.

Table 33—Summary of MLME primitives

Name	Request	Confirm	Indication	Response
MLME-MESH-CAPABILITY	6.3.18.1	6.3.18.2	6.3.18.3	–

6. 3.6 Disassociation from a piconet

Change and insert the following rows in Table 3h as shown. The other rows are unchanged and are not shown.

Table 3h—MLME-DISASSOCIATE primitive parameters

Name	Type	Valid range	Description
TrgtPNID	Integer	0-65535	The PNID of the piconet to be disassociated
OrigPNID	Integer	0-65535	The PNID of the disassociated piconet
OrigDEVID	Integer	Any valid DEVID, as defined in 7.2.3	The DEVID of the device to be disassociated

6.3.6.1 MLME-DISASSOCIATE.request

Change the primitive definition in 6.3.6.1 as shown:

```
MLME-DISASSOCIATE.request       (
                                TrgtPNID,
                                OrigDEVID,
                                Timeout
                                )
```

6.3.6.3 MLME-DISASSOCIATE.indication

Change the first paragraph in 6.3.6.3 as shown:

```
MLME-DISASSOCIATE.indication    (
                                OrigPNID,
                                DEVID,
                                DEVAddress,
                                ResultCode,
                                ReasonCode
                                )
```

Insert the following paragraph after 6.3.17 in as shown:

6.3.18 Mesh capability information management

The following primitives are used that an MPNC requests add the mesh capability information in the beacon. The parameters used for these primitives are defined in Table 3x

Table 3x— MLME-MESH-CAPABILITY primitive parameters

Name	Type	Valid range	Description
MeshID	Integer	0-65535	The identity of a specific mesh network
HopCountToMC	Integer	0–MESHPIB_MaxTreeLevel	A hop distance from the current MPNC to the MC
StartingTREEID	Integer	0–65535	The lowest TREEID of the TREEID block, which is assigned by the parent MPNC
EndingTREEID	Integer	0–65535	The highest TREEID of the TREEID block, which is assigned by the parent MPNC
ResultCode	Enumeration	SUCCESS, FAILURE	Indicates the result of the TREEID request.
ReasonCode	Enumeration	NO_TREEID_BLOCK_AVAILABLE, NO_CTA_AVAILABLE, NO_CHILD_MPNC_SUPPORT	Indicates the reason for the failure of primitive

6.3.18.1 MLME-MESH-CAPABILITY.request

This primitive requests an MPNC adds the mesh capability information in its beacon. The semantics of this primitive are:

```
MLME-MESH-CAPABILITY.request    (
                                MeshID,
                                HopCountToMC,
                                SrartingTREEID,
                                EndingTREEID
                                )
```

The primitive parameter is defined in Table 3x.

6.3.18.2 MLME-MESH-CAPABILITY.confirm

This primitive is used to confirm the result of the mesh capability information procedure. The semantics of this primitive are:

```
MHLME-MESH-CAPABILITY.confirm  (
                               ResultCode,
                               ReasonCode
                               )
```

The primitive parameters are defined in Table 3x.

6.3.18.3 MLME-MESH-CAPABILITY.indication

This primitive reports the reception of the mesh capability information from other MPNCs in the mesh network. The semantics of this primitive are:

```
MHLME-MESH-CAPABILITY.indication (
                                 MeshID,
                                 HopCountToMC,
                                 SrartingTREEID,
                                 EndingTREEID
                                 )
```

The primitive parameters are defined in Table 3x.

6.5 MAC management

6.5.1 MAC PIB group

Change the following low in Table 33 as shown. The other rows are unchanged and not shown.

Table 33: MAC PIB PNC group parameters (continued)

Managed object	Otects	Definition	Access
MACPIB_MeshPNCCapable	1 bit	If the bit set to one, the PNC is mesh capable	Read
MACPIB_NumPNIDSet	1	The number of PNID set	Read/write
MACPIB_PNIDSet	Set	This attribute indicates the PNID of the piconets that are currently associated with. For every PNID, there exists a corresponding DEVID assigned by the destination MPNC.	Read/write

6.5.2 MAC PIB PNC group

Insert the following rows in Table 34as shown. The other Rows are unchanged and are not shown.

Table 34—MAC PIB implementation group parameters

Managed object	Octets	Definition
MESHPIB_MeshDEVCapable	1 bit	If the bit set to one, the DEV is mesh capable

6.6 MAC SAP

Change and insert the following rows in Table 36 as shown. The other rows are unchanged and are not shown.

Table 36— MAC-ASYNC-DATA and MAC-ISOCH Primitive Parameters

Name	Type	Valid range	Description
TrgtPNID	Integer	Any valid PNID, as defined in 7.2.2 of IEEE Std 802.15.3-2003.	Specifies the target PNID of an MSDU.
OrigPNID	Integer	Any valid PNID, as defined in 7.2.2 of IEEE Std 802.15.3-2003.	Specifies the originator PNID of an MSDU.
MeshHeaderPresent	Boolean	TRUE, FALSE	TRUE indicates that mesh sublayer header is present in the dataframe.
TrgtID	Integer	Any valid DEVID, asdefined in 7.2.3 of IEEE Std 802.15.3b-2005.	Specifies the target DEVID of an MSDU.
OrigID	Integer	Any valid DEVID, asdefined in 7.2.3 of IEEE Std 802.15.3b-2005.	Specifies the originator DEVID of an MSDU.

6.6.1 MAC-ASYNC-DATA.request

Change the first paragraph in 6.6.1 as shown:

This primitive is used to initiate the transfer of an MSDU from one MAC entity to another MAC entity or entities. If MeshHeaderPresent is TRUE, then the MAC will use the mesh dataframe type for transmission. If SNAPHeaderPresent is TRUE, then the MAC will use the SNAP dataframe type for transmission. All asynchronous data should use LLC/SNAP headers to avoid interoperabilityproblems in piconets where different FCSLs are being used. The semantics of this primitive are:

```
MAC-ASYNC-DATA.request  (
                        RequestID,
                        TrgtPNID,
                        TrgtID,
                        OrigID
                        TransmitTimeout,
                        MaxRetries,
                        MeshHeaderPresent,
                        SNAPHeaderPresent,
                        UserPriority,
                        ACKRequested,
                        ConfirmRequested,
                        Length,
                        Data
                        )
```

6.6.3 MAC-ASYNC-DATA.indication

Change the primitive definition in 6.6.3 as shown:

MAC-ASYNC-DATA.indication (
TrgtPNID,
TrgtID,
OrigPNID,
OrigID,
MeshHeaderPresent,
SNAPHeaderPresent,
Length,
Data
)

6.6.4 MAC-ISOCH-DATA.request

Change the primitive definition in 6.6.4 as shown:

MAC-ISOCH-DATA.request (
RequestID,
StreamIndex,
TransmitTimeout,
MaxRetries,
MeshHeaderPresent,
SNAPHeaderPresent,
ACKRequested,
ConfirmRequested,
Length,
Data
)

6.6.6 MAC-ISOCH-DATA.indication

Change the primitive definition in 6.6.6 as shown:

MAC-ISOCH-DATA.indication (
TrgtPNID,
TrgtID,
OrigPNID,
OrigID,
StreamIndex,
MeshHeaderPresent,
SNAPHeaderPresent,
Length,
Data
)

7. Frame format

7.2.1.2 Frame Type

Change and insert the following rows in Table 39 as shown. The other rows are unchanged and are not shown.

Table 39—Valid frame type values (numeric values in this table are shown in binary)

Type value b_5 b_4 b_3	Frame type description	Subclause
110	Mesh data frame	7.3.6
111	reserved	

After 7.3.5.2, insert the following subclauses as 7.3.6, 7.3.6.1, 7.3.6.2:

7.3.6 Mesh data frame

7.3.6.1 Non-secure mesh data frame

The Non-secure Mesh Data frame is identical to a Non-secure Data frame, as defined in 7.3.4.1, except that the Data Payload field includes a mesh header as the first octets in the payload. The mesh header may or maynot be followed by the LLC/SNAP header. The size of the combination of the data and mesh header should not exceed the limits for the Data Payload field, as defined in 7.3.4.1. A Non-secure Mesh Data frame should be formatted as illustrated in Figure 22.

The frame type should be set to the mesh data frame value in Table 39 and the SEC bit should be set to zero. The other fields in the MAC header take on values that are appropriate for that particular data frame. All fields in the MAC header of a Non-secure Mesh Data frame should be decoded on reception.

7.3.6.2 Secure mesh data frame

The Secure Mesh Data frame is identical to a Secure Data frame, as defined in 7.3.4.2, except that the Data Payload field includes a mesh header as the first octets in the payload. The mesh header may or maynot be followed by the LLC/SNAP header. The size of the combination of the data and mesh header should not exceed the limits for the Data Payload field, as defined in 7.3.4.1. A Secure Mesh Data frame should be formatted as illustrated in Figure 23.

The frame type should be set to the mesh data frame value in Table 39 and the SEC bit should be set to one. The other fields in the MAC header take on values that are appropriate for that particular data frame. All fields in the MAC header of a Secure Mesh Data frame should be decoded on reception.

7.4 Information elements

Change and insert the following rows in Table 48 as shown. The other rows are unchanged and are not shown.

Table 48—Information elements

Element ID hex value	Element	Subclause	Present in beacon
0x14	Mesh capability	7.4.22	In every beacon of MPNC
0x15~~0x10~~-0x7F	Reserved		

7.4.11 Capability

Replace Figure 41 with the following:

bits:b7	b6	b5	b4	b3	b2	b1	b0
PNC capable	PNC Des-mode	SEC	PSRC	Next PNC capable	Reserved		Mesh capable

Figure41—PNC rating field format

Insert the following paragraph after Figure 41 in 7.4.11:
The Mesh Capable bit should be set to one if the PNC is capable of performing the mesh functionality. Otherwise the bit should be set to zero.

Replace Figure 42 with the following:

bits:b7	b6	b5	b4	b3	b2	b1	b0
Preferred fragment size			Supported data rates				

bits:b15	b14	b13	b12	b11	b10	b9	b8
Reserved	STP	CTA relinquish	Imp-ACK	Dly-ACK	Listen to Multicast	Listen to Source	Always AWAKE

bits:b23	b22	b21	b20	b19	b18	b17	b16
Reserved							Mesh capable

Figure42—DEV capabilities field format

Insert the following paragraphs after the seventeenth paragraph in 7.4.11:

The Mesh Capable bit should be set to one if the DEV is capable of performing the mesh functionality. Otherwise the bit should be set to zero.

Insert the following subclause 7.4.22 after 7.4.21:

7.4.22 Mesh Capability

The Mesh Capability IE is used by the MPNC to inform the MPNCs of certain capability of a mesh enabled network. The Mesh Capability IE should be formatted as illustrated in Figure 48.

octet: 2	**2**	**1**	**2**	**1**	**1**
Ending TREEID	Starting TREEID	Hop count to MC	Mesh ID	Length	Element ID

Figure 48f— Mesh capability information element format

The Mesh ID field contains the unique identifier for the mesh network. The mesh ID normally remains constant during the current instantiation of the mesh network and may be persistent for multiple sequential instantiations of the piconet by the same MPNC

The Hop Count To MC filed indicates the number of hop count to the MC.

The Starting TREEID field indicates the smallest TREEID of the TREEID block which is assigned by the parent MPNC.

The Ending TREEID field indicates the largest TREEID of the TREEID block which is assigned by the MC.