

INTERNATIONAL
STANDARD

ISO/
IEC/IEEE
8802-1CS

First edition
2022-07

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

**Part 1CS:
Link-local registration protocol**

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

Partie 1CS: Protocole d'enregistrement de type liaison locale



Reference number
ISO/IEC/IEEE 8802-1CS:2022(E)

Documents
Document Sharing Hub
© IEEE 2021

Authorized licensed use limited to: Tsinghua University. Downloaded on February 26, 2026 at 08:12:00 UTC from IEEE Xplore. Restrictions apply.



COPYRIGHT PROTECTED DOCUMENT

© IEEE 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from IEEE at the address below.

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

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

Published in Switzerland

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO/IEC documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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

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

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

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

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

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

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

IEEE Standard for Local and Metropolitan Area Networks— Link-local Registration Protocol

Developed by the

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 3 December 2020

IEEE SA Standards Board

Abstract: Protocols, procedures, and managed objects for a Link-local Registration Protocol (LRP) to replicate a registration database from one end to the other of a point-to-point link and to replicate changes to parts of that database are specified in this standard. A facility is provided to purge the replicated database if the source becomes unresponsive. LRP is optimized for databases on the order of 1 Mbyte.

Keywords: Bridged Local Area Networks, bridges, bridging, IEEE 802[®], IEEE 802.1CS[™], IEEE 802.1Q[™], Link-local Registration Protocol, local area networks (LANs), LRP, MAC Bridges, Time-Sensitive Networking, TSN, Virtual Bridged Local Area Networks (virtual LANs)

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

Copyright © 2021 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 23 April 2021. Printed in the United States of America.

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

OMG, UML, and Unified Modeling Language are either registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.

PDF: ISBN 978-1-5044-7253-1 STD24533
Print: ISBN 978-1-5044-7254-8 STDPD24533

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

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

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

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

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

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

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

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

Translations

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

Official statements

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

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and Standards Coordinating Committees are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#). An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute 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.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

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

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

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

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#). For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#). Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the [IEEE SA Patent Policy](#).

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at <https://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

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

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Participants

At the time this standard was submitted to the IEEE SA for approval, the Higher Layer LAN Protocols Working Group had the following membership:

Glenn Parsons, *Chair*
Jessy V. Rouyer, *Vice Chair*
Norman Finn, *Editor*

Astrit Ademaj
 Ralf Assmann
 Christian Boiger
 Paul Bottorff
 Radhakrishna Canchi
 Feng Chen
 Weiying Cheng
 Paul Congdon
 Rodney Cummings
 Josef Dorr
 Hesham Elbakoury
 Thomas Enzinger
 János Farkas
 Donald Fedyk
 Geoffrey Garner
 Craig Gunther
 Marina Gutierrez
 Stephen Haddock
 Ruibo Han
 Mark Hantel
 Jerome Henry
 Marc Holness

Satoko Itaya
 Yoshihiro Ito
 Michael Karl
 Stephan Kehrer
 Randy Kelsey
 Hajime Koto
 James Lawlis
 Christophe Mangin
 Scott Mansfield
 Kenichi Maruhashi
 David McCall
 Larry McMillan
 John Messenger
 Hiroki Nakano
 Bob Noseworthy
 Tomoki Ohsawa
 Hiroshi Ohue
 Donald R. Pannell
 Michael Potts
 Dieter Proell
 Wei Qiu

Karen Randall
 Maximilian Riegel
 Atsushi Sato
 Frank Schewe
 Michael Seaman
 Maik Seewald
 Ramesh Sivakolundu
 Johannes Specht
 Marius Stanica
 Guenter Steindl
 Karim Traore
 Balazs Varga
 Tongtong Wang
 Hao Wang
 Karl Weber
 Ludwig Winkel
 Jordon Woods
 Takahiro Yamaura
 Xiang Yu
 Nader Zein
 William Zhao
 Helge Zinner

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

Thomas Alexander
 Amelia Andersdotter
 Christian Boiger
 Vern Brethour
 William Byrd
 Paul Cardinal
 Steven Carlson
 Pin Chang
 Jin Seek Choi
 Paul Congdon
 Charles Cook
 Rodney Cummings
 Norman Finn
 Avraham Freedman
 Matthias Fritsche
 Stephen Haddock
 Marco Hernandez
 Werner Hoelzl
 Russell Housley

Yasuhiro Hyakutake
 Raj Jain
 Pranav Jha
 Peter Jones
 Piotr Karocki
 Stephan Kehrer
 Randy Kelsey
 Stuart Kerry
 Evgeny Khorov
 Yongbum Kim
 Hyeong Ho Lee
 James Lepp
 Christophe Mangin
 Arthur Marris
 Jonathon McLendon
 Nick S. A. Nikjoo
 Paul Nikolich
 Satoshi Obara
 Robert O'Hara

Bansi Patel
 Clinton Powell
 Dieter Proell
 R. K. Rannow
 Maximilian Riegel
 Jessy V. Rouyer
 Peter Saunderson
 Frank Schewe
 Manikantan Srinivasan
 Walter Struppler
 Mitsutoshi Sugawara
 Mark-Rene Uchida
 John Vergis
 George Vlantis
 Lisa Ward
 Karl Weber
 Scott Willy
 Yu Yuan
 Oren Yuen

When the IEEE SA Standards Board approved this standard on 3 December 2020, it had the following membership:

Gary Hoffman, *Chair*
Jon Walter Rosdahl, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Ted Burse
Doug Edwards
J. Travis Griffith
Grace Gu
Guido R. Hiertz
Joseph L. Koepfinger*

David J. Law
Howard Li
Dong Liu
Kevin Lu
Paul Nikolich
Damir Novosel
Dorothy Stanley

Mehmet Ulema
Lei Wang
Sha Wei
Philip B. Winston
Daidi Zhong
Jingyi Zhou

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 802.1CS-2020, IEEE Standard for Local and Metropolitan Area Networks—Link-local Registration Protocol.

This standard defines the Link-local Registration Protocol.

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

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

Contents

1.	Overview	13
1.1	Scope	13
1.2	Purpose	13
1.3	State diagram conventions	13
1.4	Specification model	13
1.5	Note on inter-table references	14
1.6	Specification precedence	14
1.7	Introduction	14
2.	Normative references	15
3.	Definitions	17
4.	Abbreviations	19
5.	Conformance	20
5.1	Introduction	20
5.2	Requirements terminology	20
5.3	Common Native and Proxy required behaviors	20
5.4	Native end system required behaviors	21
5.5	Native end system optional behaviors	21
5.6	Native relay system required behaviors	21
5.7	Native relay system optional behaviors	21
5.8	Proxy system required behaviors	22
5.9	Proxy system optional behaviors	22
5.10	Controlled system optional behaviors	22
6.	Link-local Registration Protocol	23
6.1	Introduction	23
6.2	Overview	23
6.3	Objectives and non-objectives	24
6.4	Proxy systems, Controlled systems, and target ports	25
6.5	Link Layer Discovery Protocol and target ports	28
6.6	Exploratory Hello LRPDU	29
6.7	Target port state	29
6.8	LRP database structure	30
6.9	LRP-DT data transport mechanisms	30
6.10	LRP-DS database synchronization mechanism	32
6.11	State machine creation	34
6.12	Security considerations	35
7.	LRP-DT Procedures	36
7.1	Introduction	36
7.2	LRP-DT instance maintenance	36
7.3	LRP-DT instance	41
8.	Portal	48
8.1	Introduction	48
8.2	Portal association maintenance	48

8.3	Applicant	54
8.4	Registrar	58
9.	Format and encoding of LRP Data Units	62
9.1	Introduction	62
9.2	AppId	62
9.3	LRP database synchronization protocol	62
9.4	LRPDU formats	63
9.5	LRP data transport protocols	69
10.	LRP-DS service interface	70
10.1	Introduction	70
10.2	Association primitives	71
10.3	Portal interface	74
11.	Managed objects	76
11.1	Introduction	76
11.2	Managed objects UML	76
11.3	System global managed objects	77
11.4	Per-LRP-DT instance managed objects	77
11.5	Per-Portal managed objects	78
11.6	LRP LLDP TLV managed objects	80
12.	YANG models for LRP	82
12.1	Introduction	82
12.2	The YANG framework	82
12.3	Security considerations	82
12.4	Relationship to other YANG modules	82
12.5	YANG data scheme definition	82
12.6	Definition of LRP YANG module	83
13.	MIB modules for LRP	92
13.1	Internet standard management framework	92
13.2	Structure of the LRP MIB	92
13.3	Relationship to the LLDP-V2-TC-MIB	93
13.4	Security considerations	93
13.5	MIB modules	96
	Annex A (normative) Protocol Implementation Conformance Statement (PICS) proforma	130
A.1	Introduction	130
A.2	PICS proforma for Link-local Registration Protocol	133
	Annex B (normative) LRP application specifications	138
B.1	Overview	138
B.2	LRP-DS versus IS-IS: LRP application responsibilities	138
B.3	Use of LLDP	139
B.4	Portal creation	139
B.5	Database locking issues	140
B.6	ECP vs. TCP issues	140
B.7	TCP active/passive OPEN	141

B.8 Resetting the databases 141

B.9 Application-level acknowledgments 141

B.10 Sequence number wrap-around 142

B.11 LRP applications’ requirements on LRP 142

Annex C (normative) IEEE 802.1 Organizationally Specific TLVs for LLDP..... 143

 C.1 Overview 143

 C.2 Organizationally Specific TLV definitions 143

Annex D (informative) Bibliography 148

IEEE Standard for Local and Metropolitan Area Networks— Link-local Registration Protocol

1. Overview

1.1 Scope

This standard specifies protocols, procedures, and managed objects for a Link-local Registration Protocol (LRP) to replicate a registration database from one end to the other of a point-to-point link and to replicate changes to parts of that database. A facility is provided to purge the replicated database if the source becomes unresponsive. Provision is made for a proxy system to operate LRP on behalf of a controlled system. LRP is optimized for databases on the order of 1 Mbyte.

1.2 Purpose

LRP is designed to facilitate the creation of applications that distribute information through all or part of a network.

1.3 State diagram conventions

This document uses the state diagram conventions defined in Annex E of IEEE Std 802.1Q-2018.¹ The programming language C (ISO/IEC 9899:2018) is also used to document the operation of conformant systems. C functions are distinguished with `this special fixed-width font (e.g., 9.4.6)`.

1.4 Specification model

The model of operation documented by this standard is simply a basis for describing the functionality of a compliant equipment. Implementations can adopt any internal model of operation compatible with the externally visible behavior that this standard specifies. Conformance of equipment to this standard is purely in respect of observable protocol.

¹ Information on references can be found in Clause 2.

1.5 Note on inter-table references

A number of state machine variables, managed objects, and interface primitives in this standard use the phrase “a reference to *xyz*” (see e.g., 7.3.2.8). Such a reference might be implemented by a pointer, an integer index into an array, the value of some other object in *xyz* that is sufficiently unique to make the reference unambiguous. Other techniques are also possible. This is an implementation choice and is not specified by this standard. This use of “a reference to” is also used for managed objects in Clause 11. The actual data representations of references to managed objects in Clause 12 and Clause 13, of course, are not arbitrary.

1.6 Specification precedence

If any conflict among parts of this standard become apparent, state machine diagrams (see 1.3) take precedence over other parts of the standard, followed by information in normative tables, followed by that in normative text, followed by that in normative figures, followed by YANG and MIB modules. Non-normative tables, figures, and text are in annexes and are clearly marked as such.

1.7 Introduction

The Link-local Registration Protocol (LRP) provides an LRP Database Synchronization (LRP-DS) service interface (Clause 10) that can be used by one or more LRP applications in a network, each of which needs to distribute information to some or all of the relay systems and end systems in a network. The information is passed hop-by-hop, in the sense that each information element is associated with a specific pair of connected ports in the network.

LRP accomplishes its task by creating point-to-point bi-directional associations between systems' application instances, each association consisting of two one-way paths, and each path consisting of an applicant database at one end, and a registrar database at the other. LRP quickly and reliably replicates each applicant database to its neighbor's registrar database. Locally to a system, the application instance performs any propagation or modification of information among the applicant and registrar databases on different ports within a system. Globally, the application instances plus LRP can propagate information throughout a network. This standard does not specify any LRP application that might use LRP. These basic relationships are illustrated in Figure 1-1.

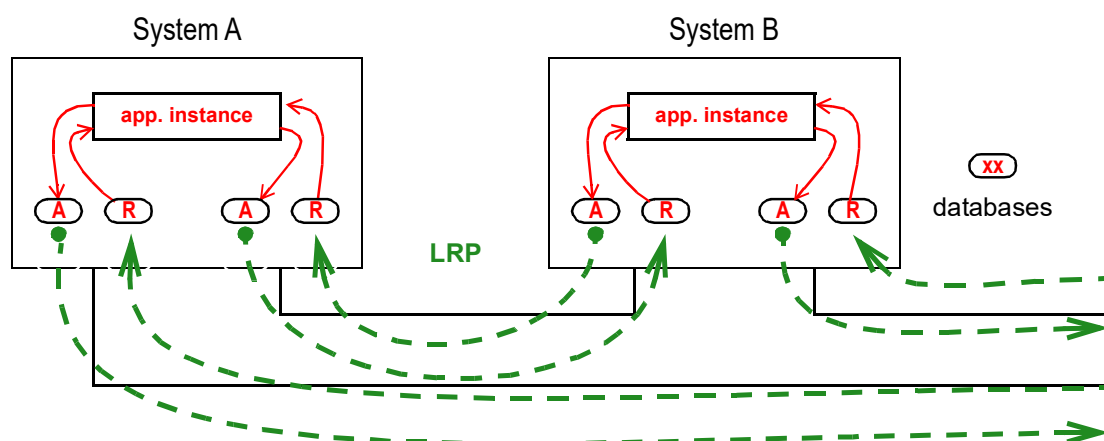


Figure 1-1—LRP simplified

LRP also supports a mode where a Proxy system operates the application instance, databases, and LRP on behalf of a Controlled system.

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.

Non-normative references (i.e., that provide additional information not required for the application of this document) are given in Annex D.

NOTE—The inclusion of a document in this list of normative references indicates that information in that document is necessary to implement the present standard. It does not imply that any other part of that referenced document is required to be implemented by a system conformant to the present standard.²

IEEE Std 802[®], IEEE Standard for Local and metropolitan area networks—Overview and Architecture.^{3, 4}

IEEE Std 802.1AB[™]-2016, IEEE Standard for Local and metropolitan area networks—Station and Media Access Control Connectivity Discovery.

IEEE Std 802.1Q[™], IEEE Standard for Local and metropolitan area networks—Bridges and Bridged Networks.

IEEE Std 1003.1[™], IEEE Standard for Information Technology—Portable Operating System Interface (POSIX[™]) Base Specifications, Issue 7.

IETF RFC 793, Transmission Control Protocol, September 1981.⁵

IETF RFC 2578, Std 58, Structure of Management Information Version 2 (SMIv2), April 1999.

IETF RFC 2579, Std 58, Textual Conventions for SMIv2, April 1999.

IETF RFC 2580, Std 58, Conformance Statements for SMIv2, April 1999.

IETF RFC 2863, The Interfaces Group MIB, June 2000.

IETF RFC 3232, Assigned Numbers: RFC 1700 is Replaced by an On-line Database, January 2002.

IETF RFC 4001, Textual Conventions for Internet Network Addresses, February 2005.

IETF RFC 6335, Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry, August 2011.

IETF RFC 7950, The YANG 1.1 Data Modeling Language, August 2016.

Internet Assigned Numbers Authority (IANA) database, Service Name and Transport Protocol Port Number Registry.⁶

² Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

³ IEEE publications are available from The Institute of Electrical and Electronics Engineers (<https://standards.ieee.org/>).

⁴ The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

⁵ IETF documents (i.e., RFCs) are available for download at <http://www.rfc-archive.org/>.

⁶ The IANA database is available at <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.

ISO/IEC 8473-1:1998, Information technology — Protocol for providing the connectionless-mode network service: Protocol specification.⁷

ISO/IEC 9899:2018, Information technology—Programming languages—C.

OMG[®] Unified Modeling Language[™] (OMG UML[®]), Version 2.5, March 2015.^{8, 9}

⁷ISO publications are available from the International Organization for Standardization (<https://www.iso.org/>) and the American National Standards Institute (<https://www.ansi.org/>). IEC publications are available from the International Electrotechnical Commission (<https://www.iec.ch>) and the American National Standards Institute (<https://www.ansi.org/>).

⁸OMG publications are available from the Object Management Group, at <http://www.omg.org>.

⁹OMG, UML, and Unified Modeling Language are either registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.

3. Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.¹⁰

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

- end station
- frame

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

- Edge Control Protocol (ECP)

The following terms are specific to this standard:

applicant: One of the two roles (with registrar) an application instance can have with respect to a Portal. The applicant controls a database that the Link-local Registration Protocol (LRP) replicates to the registrar in the neighbor Portal.

application instance: An instance of a Link-local Registration Protocol (LRP) application running in a particular Native or Proxy system.

Controlled system: An end system or relay system that has one or more target ports, but not their associated Portals.

Edge Control Protocol (ECP): A protocol that provides reliable delivery of control service data units (SDUs). (From IEEE Std 802.1Q)

Edge Control Protocol Data Unit (ECPDU): The basic unit of information transmitted or received by the Edge Control Protocol.

end station: A functional unit in an IEEE 802 network that acts as a source of, and/or destination for, link layer data traffic carried on the network. (From IEEE Std 802)

end system: A system attached to a network that is an initial source or a final destination of data transmitted across that network.

NOTE—The term “end system” is often used in this document in places where the reader of IEEE 802 standards would expect the term “end station,” in order to avoid confusion caused by standards relating to routers. For example, a router, as defined by IETF, is an IEEE 802 “end station,” but not an “end system,” because it is not typically an initial source or final destination. Where this standard specifically refers to the use of IEEE 802 services, the term “end station” is used. Where it refers to more generalized instances of connectionless services, the term “end system” is used.

frame: The format of aggregated bits from a medium access control (MAC) sublayer entity that are transmitted together in time. (From IEEE Std 802)

local target port: A target port within the same Native system as an application instance using the port, or a target port within a Controlled system serviced by an application instance in a Proxy system using the port.

¹⁰*IEEE Standards Dictionary Online* is available at <https://dictionary.ieee.org>. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

LRP application: A protocol, identified by an application ID, that makes use of the Link-local Registration Protocol (LRP).

Native system: An end system or relay system that has one or more target ports and their associated Portals.

Neighbor Portal: The one Portal to which a Portal has been connected.

neighbor target port: A target port in a system directly connected to a local port.

port: An entity defined by a {Chassis ID, Port ID} pair that is unique over the union of systems interconnected by the Link-layer Registration Protocol (LRP). A port can be tied to a Service Access Port (SAP) providing access to an instance of the MAC Service described in IEEE Std 802.1AC.

NOTE—See IEEE Std 802.1AC™ [B3].¹¹

Portal: An instantiation of the Portal interface, along with instances of the applicant and/or registrar state machines and databases, associated with a single application instance, residing in a Native or Proxy system, and associated with a target port.

Proxy system: A system that has Portals, but not the associated target ports.

record: A subset of a database that is transferred as a single unit from the applicant to the registrar by the Link-local Registration Protocol (LRP).

registrar: One of the two roles (with applicant) an application instance can have with respect to a Portal. The registrar receives the copy of the database that the Link-local Registration Protocol (LRP) replicates from the applicant in the neighbor Portal.

relay system: A router or a bridge.

NOTE—The term “relay system” is often used in this document in places where the reader of IEEE 802 standards would expect the term “bridge.” A relay system can, in theory, be a router, a bridge, or any other device that forwards a frame from one port to another, such that the frame is, in some useful sense, identifiable to other systems as being the same frame. Where this standard specifically refers to one or the other, the terms “router” or “bridge” are used. Where it refers to more generalized instances of connectionless services, the term “relay system” is used.

system: An end system or a relay system.

target port: A port in a Native or Controlled system with which a Portal is associated.

¹¹The numbers in brackets correspond to those of the bibliography in Annex D.

4. Abbreviations

This standard contains the following abbreviations:

CID	Company ID
ECP	Edge Control Protocol
ECPDU	Edge Control Protocol Data Unit
LLDP	Link Layer Discovery Protocol (IEEE Std 802.1AB)
LLDPDU	Link Layer Discovery Protocol Data Unit
LRP	Link-local Registration Protocol
LRP-DS	Link-local Registration Protocol–Database Synchronization
LRP-DT	Link-local Registration Protocol–Data Transport
LRPDU	Link-local Registration Protocol Data Units
OUI	Organizationally Unique Identifier
TCP	Transmission Control Protocol
TLV	Type Length Value
UML	Unified Modeling Language

5. Conformance

5.1 Introduction

This clause specifies the mandatory and optional behaviors of conformant implementations of this standard. Following a subclause on Requirements terminology (5.2), four types of systems are defined in this clause:

- a) The behaviors for Native end systems are specified in 5.3, 5.4 (mandatory) and 5.5 (optional).
- b) The behaviors for Native relay systems are specified in 5.3, 5.6 (mandatory) and 5.7 (optional).
- c) The behaviors for Proxy systems are specified in 5.3, 5.8 (mandatory) and 5.9 (optional).
- d) The optional behaviors for Controlled systems are specified in 5.10.

5.2 Requirements terminology

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

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

The Protocol Implementation Conformance Statement (PICS) proformas (see Annex A) reflect the occurrences of the words “shall,” “may,” and “should” within the standard.

The standard avoids needless repetition and apparent duplication of its formal requirements by using *is*, *is not*, *are*, and *are not* for definitions and the logical consequences of conformant behavior. Behavior that is permitted but is neither always required nor directly controlled by an implementor or administrator, or whose conformance requirement is detailed elsewhere, is described by *can*. Behavior that never occurs in a conformant implementation or system of conformant implementations is described by *cannot*. The word *allow* is used as a replacement for the phrase “support the ability for,” and the word *capability* means “can be configured to.”

5.3 Common Native and Proxy required behaviors

A Native or Proxy system conformant to this standard shall:

NOTE—The specification of a particular LRP application can impose further constraints on LRP behavior.

- a) Provide the LRP-DT instance state machine, as specified in 7.2 and 7.3;
- b) Provide the Portal association maintenance state machines, as specified in 8.2;
- c) Provide at least one of the Link-local Registration Protocol Data Transport mechanisms, Edge Control Protocol (ECP) or Transmission Control Protocol (TCP), as specified in 6.9.1 and 6.9.2;
- d) Conform to the specifications for the format and encoding of Link-local Registration Protocol Data Units (LRPDUs) in Clause 9; and
- e) Provide access to the managed objects required for the functions in item a in this subclause, according to the specifications in Clause 11.

5.4 Native end system required behaviors

A Native end system conformant to this standard shall, on one or more ports:

- a) Conform to the required common capabilities specified in 5.3; and
- b) Provide either the Applicant state machine as specified in 8.3, the Registrar state machines, as specified in 8.4, or both.

5.5 Native end system optional behaviors

A Native end system conformant to this standard that provides the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) should:

- a) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for the LRP TCP Discovery TLV specified in C.2.2.

A Native end system conformant to this standard that does not provide the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) may:

- b) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for LLDP TLVs specified in Annex C.

A Native end system conformant to this standard may:

- c) Provide access to the managed objects required by 5.3 and 5.4 or selected from 5.5 using the YANG modules specified in Clause 12; and/or
- d) Provide access to the managed objects required by 5.3 and 5.4 or selected from 5.5 using the MIB modules specified in Clause 13.

NOTE—The specification of a particular LRP application can make mandatory any of the optional behaviors specified in this clause (see B.11).

5.6 Native relay system required behaviors

A Native relay system conformant to this standard shall, on more than one port:

- a) Conform to the required common capabilities specified in 5.3; and
- b) Provide all of the Applicant and Registrar state machines, as specified in 8.3 and 8.4.

5.7 Native relay system optional behaviors

A Native relay system conformant to this standard that provides the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) should:

- a) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for the LRP TCP Discovery TLV specified in C.2.2.

A Native relay system conformant to this standard that does not provide the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) may:

- b) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for LLDP TLVs specified in Annex C.

A Native relay system conformant to this standard may:

- c) Provide access to the managed objects required by 5.3 and 5.6 or selected from 5.7 using the YANG modules specified in Clause 12; and/or
- d) Provide access to the managed objects required by 5.3 and 5.6 or selected from 5.7 using the MIB modules specified in Clause 13.

NOTE—The specification of a particular LRP application can make mandatory any of the optional behaviors specified in this clause (see B.11).

5.8 Proxy system required behaviors

A Proxy system conformant to this standard shall:

- a) Provide the LRP-DT instance state machine, as specified in 7.2 and 7.3;
- b) Provide the Portal association maintenance state machines, as specified in 8.2;
- c) Provide all of the Applicant and Registrar state machines, as specified in 8.3 and 8.4;
- d) Provide the LRP-DT TCP mechanism, as specified in 6.9.2;
- e) Conform to the specifications for the format and encoding of LRPDUs in Clause 9; and
- f) Provide access to the managed objects required for these functions, according to the specifications in Clause 11.

5.9 Proxy system optional behaviors

A Proxy system conformant to this standard may:

- a) Provide access to the managed objects required by 5.3 and 5.8 using the YANG modules specified in Clause 12; and/or
- b) Provide access to the managed objects required by 5.3 and 5.8 using the MIB modules specified in Clause 13.

NOTE—The specification of a particular LRP application can make mandatory any of the optional behaviors specified in this clause (see B.11).

5.10 Controlled system optional behaviors

A Controlled system, whether an end system or a relay system, should:

- a) Conform to the required behaviors of IEEE Std 802.1AB Station and Media Access Control Connectivity Discovery, and to the provisions for LLDP TLVs specified in Annex C;
- b) Provide access to the managed objects required to configure the address(es) of its Proxy system specified in 11.6.1; and/or
- c) Provide access to the managed objects for monitoring LLDP TLVs as specified in 11.6.2.

A Controlled system, whether an end system or a relay system, may:

- d) Provide access to the implemented managed objects using the YANG modules specified in Clause 12; and/or
- e) Provide access to the implemented managed objects using the MIB modules specified in Clause 13.

6. Link-local Registration Protocol

6.1 Introduction

The Link-local Registration Protocol (LRP) is described in the following subclauses:

- a) An overview of the LRP is given in 6.2.
- b) Objectives and non-objectives are given in 6.3.
- c) Descriptions of Native, Proxy, and Controlled systems are given in 6.4.
- d) The role of the Link Layer Discovery Protocol (LLDP) is given in 6.5.
- e) A description of Exploratory Hello LRPDU is given in 6.6.
- f) The handling of port state changes are given in 6.7.
- g) The structure of the LRP databases is given in 6.8.
- h) Two methods for connecting Portals with LRP Data Transport protocol (LRP-DT) are given in 6.9.
- i) A description of the LRP Database Synchronization protocol (LRP-DS), which runs over LRP-DT to actually synchronize the applicant and registrar databases, is given in 6.10.
- j) A brief description of the process of creating the LRP state machines is given in 6.11.
- k) Security considerations are given in 6.12.

6.2 Overview

A brief explanation of the purpose of LRP is given in 1.7. This overview describes the relationships among the various functions comprising LRP in more detail. The basic protocol relationships are shown in Figure 6-1.

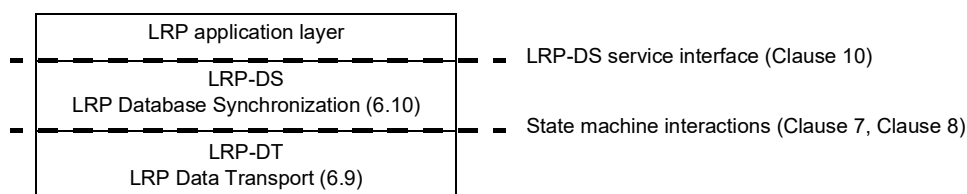


Figure 6-1—LRP protocol stack in a relay system

In Figure 6-1, the LRP application layer is the user of LRP. LRP is split into two layers: The LRP-DS database synchronization mechanism, and the LRP-DT data transport mechanisms. An application instance, operating at the LRP application layer, and a Portal operating at the LRP-DS layer, cooperate to manage applicant and registrar databases associated with target ports, as shown in Figure 1-1 and Figure 6-2. The LRP-DS Portal uses an LRP-DT instance to connect, via another system's LRP-DT instance, to a neighbor LRP-DS Portal. LRP-DT can use either the ECP or TCP to transport data. The state machines in the LRP-DS layer use the LRP-DT instances to exchange LRPDUs, by which means they copy the databases from system to system.

LRP-DT delivers data over a point-to-point association with the following services:

- a) Pacing of transmissions to avoid overflowing the receiver's buffers.
- b) Acknowledgment and retransmission of data to give some level of assurance it has been received.
- c) In-order delivery of transmissions.

- d) Repacking LRPDU into suitably sized transmission units (packets) (TCP only).
- e) Notification of a failure of a connection (failure to receive an acknowledgment) (TCP only).

Because LRP-DT provides these services, LRP-DS does not have a mechanism to retransmit data. Because ECP does not provide connection failure indications (item e, above) LRP-DS performs a periodic check (see 6.10.1) on the integrity of the database.

NOTE—Periodic integrity checks are performed, whether ECP or TCP is used for the LRP-DT.

This model is expanded in Figure 6-2, which shows the applicant and registrar databases, two systems A and B, each supporting two ports and the application instances of two different LRP applications, with both systems connected to other systems (off page to the right).

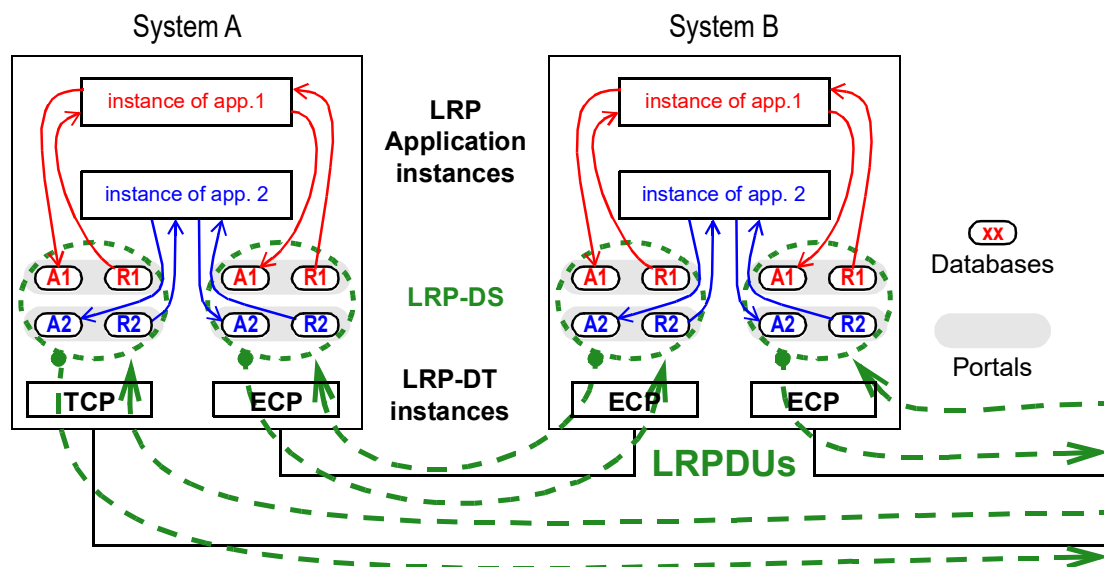


Figure 6-2—LRP application instances, databases, Portals, and LRP

6.3 Objectives and non-objectives

6.3.1 Objectives of LRP

The objectives of LRP are to:

- a) Serve application instances as described in 6.2.
- b) Remain independent of the syntax and semantics of the LRP application data contained in the applicant and registrar databases.
- c) Provide a facility for an application instance to discover its peers, and to make and break associations between application instances.
- d) Support multiple options for the transport mechanisms used to carry LRPDU, in order to provide a range of capability/complexity trade-offs.
- e) Transfer the application instances' data quickly, reliably, and efficiently with respect to bandwidth.
- f) Efficiently transfer changes to parts of a database, without retransmitting the whole database.
- g) Serve applicant and registrar databases whose size, per port, is on the order of magnitude of 1 Mbyte.

- h) Support only point-to-point associations between application instances.
- i) Support multiple point-to-point associations on one port.
- j) By the use of TCP as an LRP-DT transport mechanism, provide for placing the application instance and Portal in a separate Proxy system from its target port, in order to facilitate central control of a network.
- k) Support proxying for a Controlled system without requiring any new behaviors of the Controlled system, in order to make it possible to deploy a new LRP application without implementing an application instance in every system.
- l) Make efficient use of TCP connections.
- m) Support resource-constrained end systems.

6.3.2 Non-objectives of LRP

Certain possible objectives for LRP are explicitly excluded from the present document; if such features are required for a given LRP application, they must be provided by the LRP application specification, as follows:

- a) Support point-to-multipoint or multipoint-to-multipoint associations among Portals.
- b) Provide, in LRP, features for maintaining consistency across the various parts of a single database, e.g., locking all or a part of the database.

6.4 Proxy systems, Controlled systems, and target ports

Every Portal and its associated applicant and registrar databases are associated with a single target port. A target port can be associated with more than one Portal if those Portals serve different LRP applications, or if the Portals connect to different neighbor Portals. A target port provides access to a link that connects to one or more other target ports, typically in other systems.

There are three classes of systems relevant to this standard. The class into which a system falls is defined per LRP application; a system can be in one class for one LRP application, and in another class for another LRP application.

Native A system in which the application instances, the target ports, and the Portals all reside.

Controlled A system in which the target ports, but not the application instances and Portals, reside.

Proxy A system in which the application instances and Portals, but not the target ports, reside.

The following term is also useful:

not-controlled Either a Native or a Proxy system. That is, one in which at least the application instances and Portals reside.

Each applicant and registrar database pair shown, for example, in Figure 1-1 or Figure 6-2, is associated with a specific Portal and target port. In order to support the objective item j in 6.3.1, this standard provides for either or both of the systems connected by the link to off-load the application instance and Portal to a separate Proxy system. The system offloading its application instance and Portal is called a *Controlled system*.

NOTE 1—This standard does not specify any means for off-loading some, but not all, of the Portals of an LRP application in a single Controlled system to a Proxy system.

NOTE 2—A system can be simultaneously a Proxy system and a Native system for the same LRP application, if it has Portals for its own target ports and also proxies for at least one other system.

NOTE 3—The term “system” is not synonymous with “chassis” or “box”; a physical device can contain parts or all of any number of systems.

Figure 6-3 illustrates two Native systems, A and B. Each Portal resides in the same system as its target port. Note that TCP could be used in place of ECP, in Figure 6-3.

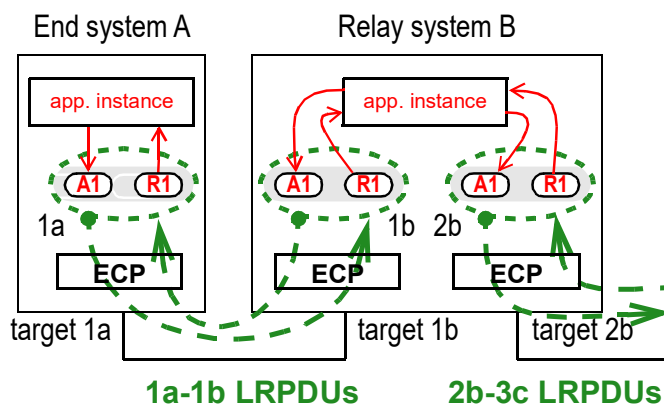


Figure 6-3—Native systems

In Figure 6-4, end system A from Figure 6-3 has been changed from a Native system to a Controlled system, and its Proxy system P has been added to the figure. As a consequence, system B must use TCP, not ECP, for its LRP-DT layer. The paths of the LRPDU are shown passing “over the top” to Proxy system P, because they are not restricted to passing through the target port. Typically, running an application instance in a Proxy system results in some change of state in the Controlled system, and vice versa. For example, a resource reservation protocol might have to adjust parameters in the queue shapers on a target port. This kind of interaction is of concern to the LRP application specification, not to this standard. It is shown in Figure 6-4 and Figure 6-5 as an arrow labeled “other control.” An existing system with no knowledge of an LRP application can be controlled by a Proxy system (item k in 6.3.1) so that it behaves as if it did understand the LRP application. All that is necessary is that the Controlled system be manageable by the Proxy system, perhaps via managed objects.

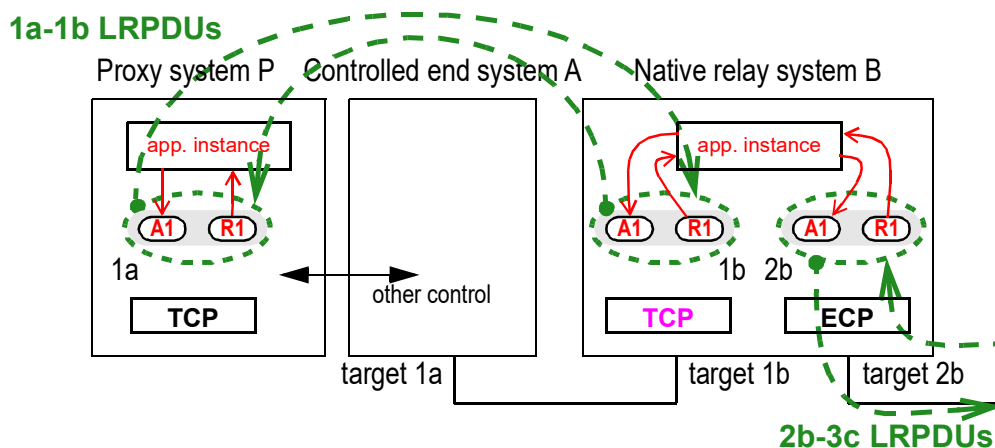


Figure 6-4—Controlled end system, Native relay systems

In Figure 6-5, we see what happens if relay system B of Figure 6-4 becomes a Controlled system to Proxy system Q. Now, the two Proxy systems exchange LRPDU for the link between target ports 1a and 1b.

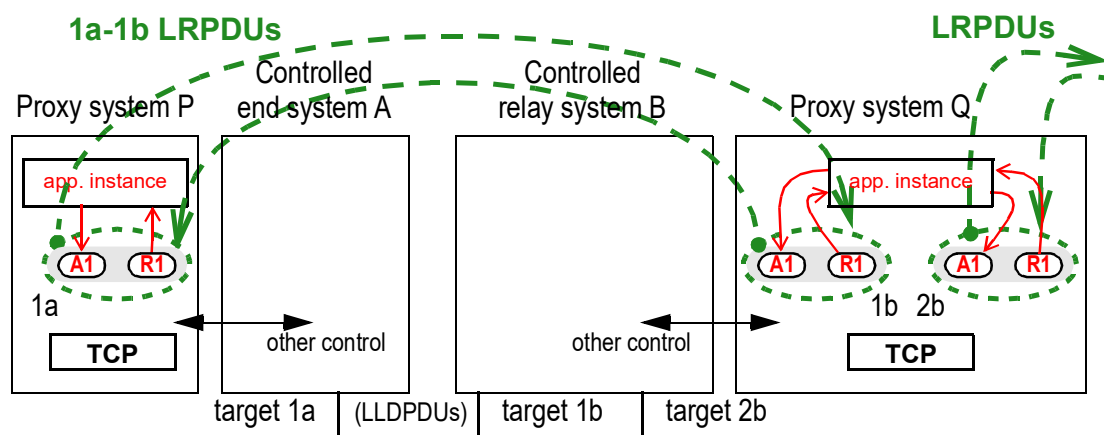


Figure 6-5—Controlled end and relay systems

As long as an LRP system uses TCP, rather than ECP, it is of no consequence to LRP or to an application instance in that system, whether its partner Portal is proxied or not. This makes it possible to design an LRP application that can be hosted either on independent cooperating relay systems, or on a centralized network controller system. Some useful examples of the use of Proxy/Controlled systems include:

- a) **Proxied end systems:** A Proxy system can proxy for a relatively simple end system, and communicate to the neighboring relay system, so that the end system does not have to implement the LRP application.
- b) **Proxied relay systems:** All relay systems in a network can use a central Proxy system. Then, a Native end system can participate in what, to it, appears to be a hop-by-hop LRP application protocol that works throughout the network, when in fact, the central Proxy system is a network controller that performs the LRP application's work in all of the relay systems in a single processor, communicating to the end systems via TCP.
- c) **End systems' proxy and relay systems' proxy:** One system can proxy for a set of cooperating Controlled end systems, while another system is a proxy for the Controlled relay systems. The two proxies can interact efficiently on behalf of their respective sets of Controlled systems. This is illustrated in Figure 6-6. In this case, no LRPDU pass between relay systems; all such traffic is internal to the network systems' proxy.

NOTE 4—TCP makes it possible to create Proxy and Controlled systems by masking whether a Native or a Proxy is exchanging LRPDU with another Native or a Proxy system. This standard does not require, and does not specify, any particular communication between the Proxy and Controlled systems.

The choice to off-load an LRP application and its Portals to a Proxy system is made independently for each LRP application. A single system can be a Controlled system for one LRP application, a Native system for another LRP application, and a Proxy system for a third LRP application.

The specification of an LRP application can choose whether to allow Proxy systems or not. If, for example, the information passed in an LRP application could prevent a system from communicating with an adjacent Controlled system's Proxy system via TCP, then support of Proxy systems by that LRP application might be inadvisable.

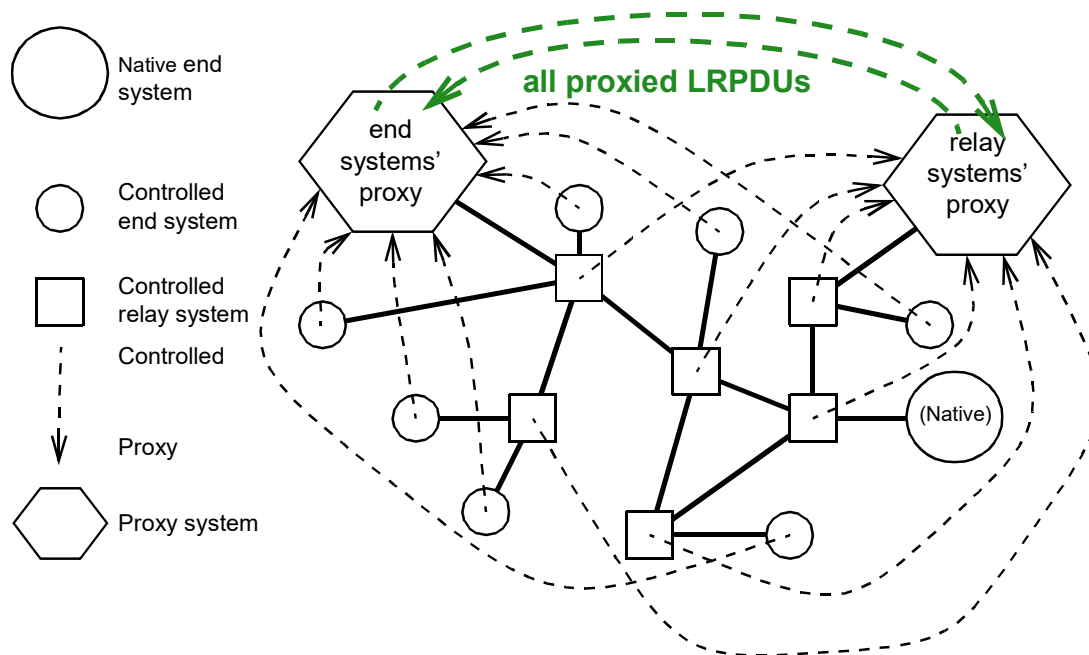


Figure 6-6—Communicating Proxy systems

6.5 Link Layer Discovery Protocol and target ports

The IEEE Std 802.1AB Link Layer Discovery Protocol (LLDP) can be used by systems to advertise and discover Portals by means of the LRP ECP Discovery TLV (C.2.1) and the LRP TCP Discovery TLV (C.2.2). When used, the LLDPDUs are sent and received through, and apply to, target ports, so only a Native or a Controlled system pass these TLVs. (A Proxy system has no target ports.) In Figure 6-5, for example, systems A and B are shown trading LLDPDUs.

If a Controlled system runs LLDP, then its Proxy system needs to access and/or control the LRP ECP Discovery TLVs and LRP TCP Discovery TLVs sent and received on the Controlled system's target ports. The means for coordinating this information is not specified in this standard, but see IEEE Std 802.1AB for a description of managed objects that make this possible.

Alternatively, the information that would be supplied via LLDP running either in a Native or a Controlled system can be configured statically in a Native system or in the Controlled's Proxy system. The managed objects required to configure this information are not defined in this standard, but could be defined in a particular application specification. (See B.3.) In addition, Exploratory Hello LRPDUs (6.6) can be used to simultaneously detect an associate with a neighboring system.

However the information about potential neighbor target ports is obtained, whether through a Native or Controlled system running LLDP, or through configuration of a Native or Proxy system, an application instance can use the Local Target Port request and Neighbor Target Port request primitives (10.2.2, 10.2.3) to convey that information to LRP.

NOTE—Running and monitoring LLDP requires more of an implementation than simply configuring the information in each system. However, running LLDP is much safer than configuration, because miswiring or misconfiguration can result in passing information between Portals about target port connections that do not, in fact, exist.

The Proxy/Controlled mechanisms defined in this standard require a meaningful way to identify the Controlled's target ports to the Proxy systems, even though IP allows the Proxy and Controlled systems to be in separate broadcast domains. The Chassis ID + Port ID mechanism of IEEE Std 802.1AB is used to identify individual target ports. An LRP application running in a network is not expected to work correctly if two or more target ports have the same Chassis ID and Port ID. It is up to those administrating the network(s) using LRP to ensure that target port identifiers are unique over an adequately large domain to meet the needs of the LRP applications deployed.

6.6 Exploratory Hello LRPDU

In some situations, support for resource-constrained systems (item m in 6.3.1) leads to a desire to not run IEEE 802.1AB LLDP in the system, but to still have some means of discovering one's neighbor target ports. The state machines in Clause 7 and Clause 8 permit the transmission and reception of an exploratory Hello LRPDU (9.4.2). This is a Hello LRPDU that carries neither the Neighbor Chassis ID TLV (9.4.2.8) nor the Neighbor Port ID TLV (9.4.2.9); it does advertise its own local target port with the My Chassis ID TLV (9.4.2.6) and the My Port ID TLV (9.4.2.7). The reception and transmission of exploratory Hello LRPDU are enabled separately (see 7.2.2.1). By this means, the system transmitting the exploratory Hello LRPDU does not need to acquire the chassis and port ID of its neighbor target port before transmitting the Hello LRPDU, either by configuration or by running LLDP.

Exploratory Hello LRPDU can only be transmitted over ECP, not over TCP, because with ECP, there is no difficulty in determining to which local target port a received exploratory Hello LRPDU is directed; a target port can have only one instance of ECP, and ECP can have only one neighbor port.

6.7 Target port state

Since each LRP Portal and its databases are tied to a specific target port, it is expected that the state of the target port, e.g., whether or not the target port has an operational physical connection, will be important to the LRP application. However, different LRP applications can have different requirements for maintaining or altering the databases and/or propagating information to the rest of the network when a target port momentarily goes operational or non-operational. Furthermore, a Proxy system has only second-hand knowledge of the state of its Controlled system's target ports. Therefore, the state of the target port has no direct impact on the LRP state machines described in Clause 7 and Clause 8.

The impact of the target port's state on the state machines can be different in different circumstances:

- a) The LRP-DT ECP mechanism (6.9.1) uses the target port to carry LRPDU, so a failed target port results in no LRPDU being sent or received, which can cause Hello LRPDU (9.4.2) to time out.
- b) The LRP-DT TCP mechanism (6.9.2) can use the target port to carry LRPDU. If so, a failed target port can cause the TCP connection (socket) to fail, which will destroy the LRP-DT instance and all its associated Portals. A failed target port can also cause Hello LRPDU (9.4.2) to time out.
- c) If the TCP connection does not pass through the target port, then a target port failure has no effect on the state machines in Clause 7 and Clause 8.
- d) In any case, the failure of a port in any system in the network through which the TCP connection passes can cause an interruption in connectivity between Portals that can cause TCP connections to fail and/or Hello LRPDU to time out.

It is up to the LRP application specification to define how to determine the state of the target port, how to convey that state to a Proxy system, and what actions to take when the state changes. The controls defined in Clause 10 allow an application instance both to receive indications of failures (Portal destruction, Hello LRPDU timeouts, or failure to synchronize the databases), and to command LRP to take action.

There are many kinds of failure of a target port; for example, a failed target port can support only one-way communication, it can have an excessive loss rate, or it can fail completely. There are many means to detect failures, including state reported by the MAC layer, the Internal Sublayer Service MAC_Operational parameter (11.2 of IEEE Std 802.1AC-2016 [B3]), or Connectivity Fault Management (Clause 18 of IEEE Std 802.1Q-2018). The best means can be different for different LRP applications; no particular means is specified in this standard.

6.8 LRP database structure

LRP imposes a small amount of structure upon the applicant and registrar databases. In particular:

- a) An applicant or registrar database is divided into one or more records.
- b) Each record has:
 - 1) An integer record identifier, unique over the records in the database;
 - 2) A sequence number. (The sequence number is used by LRP-DS only; it is not visible to the LRP application layer);
 - 3) A checksum computed over the whole record (9.4.6) that is computed by an LRP-DS Portal and exchanged in LRPDUs (The checksum is used by LRP-DS only; it is not visible to the LRP application layer); and
 - 4) One or more octets of data.
- c) A given LRP-DT instance, using a given LRP-DT transport mechanism, can impose a maximum record size on an LRP application. The format of the LRPDU (Clause 9) limits a record's data to something less than 65 535 octets. If the LRP-DT ECP mechanism (6.9.1) is used, the maximum is further reduced to something less than the maximum frame size of the physical medium.
- d) The interaction between LRP-DS and the LRP application layer, and between associated LRP-DS Portals, is defined in terms of adding, withdrawing, or altering whole records.
- e) The data portion of a record is opaque to LRP; its structure and meaning are defined by the LRP application specification.

This division into records helps meet the goals of keeping LRP independent of the LRP applications it serves (item b in 6.3.1) and also being efficient when transferring parts of a database (item f in 6.3.1). The record identifier + sequence structure provides handles to identify the elements (records) that LRP exchanges with an application instance, so that LRP can replicate changes to the database, ensure its integrity across system restarts, and acknowledge the receipt of changes, without frequent transmissions of the entire database. An application instance can divide its applicant database into the parts that are expected to be (or have proven to be) relatively static from those that are dynamic, and put them in separate records. Then, LRP can transmit dynamic changes without affecting the static part of the database, or other dynamic records that are not changing at the same time, without understanding the details of the LRP application, and without itself implementing complex heuristics.

6.9 LRP-DT data transport mechanisms

6.9.1 LRP-DT ECP mechanism

LRPDUs can be transferred using the Edge Control Protocol (ECP) described in Clause 43 of IEEE Std 802.1Q-2018. ECP runs over the target port itself, so it can only be used by a Native system. One ECPDU (one frame) can carry a number of LRPDUs, but no LRPDU can span multiple ECPDUs. This limits the size of an LRPDU, and hence the size of one record. The reason for this limitation is that, unlike TCP (6.9.2), ECP fails silently if no acknowledgment is received after some number of retries (default 3, see 12.26.1.2 of IEEE Std 802.1Q-2018). If such a failure dropped one fragment of an LRPDU, then the next

LRPDU or LRPDU fragment could not be parsed by the receiver. Limiting an LRPDU to a single ECPDU ensures that only whole LRPDUs can be lost by ECP.

The ECP `ackTimerInit` value (D.2.12.6 of IEEE Std 802.1Q-2018), shall be 20.480 ms (RTE=11). This value is visible as a managed object (`lrpAckTimerInit`, 11.3.1).

NOTE—The choice made by ECP for a value of `ackTimerInit`, when there is more than one protocol using an ECP instance, is not fully defined by IEEE Std 802.1Q-2018.

Because ECP allows only a single instance of ECP per physical port, which instance converses with only one neighbor system, ECP can have multiple point-to-point associations (item *i* in 6.3.1) only if an ECP instance is used by more than one LRP application. The associations are multiplexed by a field in the LRPDUs (My Portal Number, 9.4.2.4) to distinguish each Portal's (association's) LRPDUs.

The application instance specifies the destination MAC address for transmitted frames carrying ECPDUs (item *b:3:i* in 10.2.3.1). If the LRP ECP Discovery TLV (C.2.1) is being transmitted, this address shall be the same as the destination MAC address used for the LLDPDUs. The Nearest Bridge group address, Nearest Customer Bridge group address, and Provider Bridge group address (Table 8-1, Table 8-2, or Table 8-3 in IEEE Std 802.1Q-2018) are reasonable choices.

6.9.2 LRP-DT TCP mechanism

LRPDUs can be transferred over a TCP connection. An LRPDU can contain up to 65 535 octets of data, the maximum value of the LRPDU's TLV Length field (9.3.3).

Because TCP can support any number of LRP-DS Portal associations through one LRP-DT instance, multiple point-to-point associations (item *i* in 6.3.1) are fully supported for a single port. The associations are multiplexed by a field in the LRPDUs (My Portal Number, 9.4.2.4) to efficiently utilize the TCP connection (item *l* in 6.3.1).

The interaction between LRP-DS and a TCP connection is described in terms of sockets, as defined by IEEE Std 1003.1. There are two ways to create a TCP socket using the functions defined in IEEE Std 1003.1: Active TCP OPEN (6.9.2.1) or Passive TCP OPEN (6.9.2.2).

6.9.2.1 Active TCP OPEN

A socket can be created using an active TCP OPEN with the following steps:

- a) The IEEE 1003.1 `socket()` function creates a socket.
- b) The IEEE 1003.1 `bind()` function can optionally attach a particular local IP address to the socket, if more than one IP address is available for use.
- c) The IEEE 1003.1 `connect()` function creates a TCP connection to a particular remote system, for use by one LRP-DT instance.

6.9.2.2 Passive TCP OPEN

A socket can be created using a passive TCP OPEN with the following steps:

- a) The IEEE 1003.1 `socket()` function creates a passive socket to be used for enabling the receipt of other systems' active TCP OPENs.
- b) The IEEE 1003.1 `bind()` function is required to attach a local IP address to the socket. See 6.9.2.3 for the selection of the TCP port number parameter to be used in the IEEE 1003.1 `bind()` function.

- c) The IEEE 1003.1 `listen()` function enables the socket to field `connect()` functions issued by other systems.

These first three steps are executed once for each local IP address used by a system for the LRP-DT TCP mechanism. The socket created is not associated with any particular LRP-DT instance. The last step is performed once for each LRP-DT instance.

- d) The IEEE 1003.1 `accept()` function clones a new socket to pair with another system's `connect()` function, leaving the original passive socket available for further `accept()` functions.

6.9.2.3 LRP TCP port number

There is no fixed value for the TCP port number for LRP. The TCP port number and IP address(es) for a particular Proxy system can be advertised in the LRP TCP Discovery TLV (C.2.2) and passed to LRP from the application instance through the Neighbor Target Port request (item b:4:iii in 10.2.3.1).

NOTE—The means by which a TCP Port number is selected is not specified by this standard. When a Native system issues the IEEE 1003.1 `bind()` function, it can then advertise that local TCP Port number (selected from the Dynamic Port range specified by IETF RFC 6335 [B15]) in its LRP TCP Discovery TLVs. A Proxy system can install the selected TCP Port numbers into the LRP TCP Discovery TLVs transmitted by its Controlled systems using managed objects (`IldpV2ConfigLrpTcpControlledTcpPortNumber` in 13.5.3). If a fixed TCP port number for LRP is assigned, that number can be advertised.

6.9.3 Quality of Service considerations

LRP is expected to be used to carry resource reservation protocols on systems implementing the queuing mechanisms required for demanding real-time uses. If frames carrying LRPDUs bypass these queuing mechanisms, that are required to provide such Quality of Service, then the real-time data streams can be disrupted.

NOTE—This standard does not specify a particular priority value (11.1 of IEEE Std 802.1AC-2016 [B3]) or a DSCP value (IETF RFC 2474 [B8]) for the ECP or TCP frames or packets. Requirements vary, but a high-priority best-effort class of service is usually a good choice for frames or packets carrying LRPDUs.

6.10 LRP-DS database synchronization mechanism

6.10.1 LRP-DS Protocol operation

The use of records by LRP-DS is modeled after the use of Link State Advertisements (LSAs) in ISO/IEC 10589:2002 IS-IS [B19]. Four LRPDUs are carried between Portals to convey database information:¹²

- 1) **Hello LRPDU:** Establishes and maintains an LRP-DS association between LRP Portals.
- 2) **Record LRPDU:** A record, with record identifier, sequence number, checksum, and data.
- 3) **Partial List LRPDU:** A list of a subset of the {record identifier, sequence number, checksum} triplets (no data) in the database.
- 4) **Complete List LRPDU:** A complete list of all of the {record identifier, sequence number, checksum} triplets (no data) in the database.

A Record LRPDU is sent from the applicant to the registrar to add, change, or delete a record. A Partial List LRPDU is sent from the registrar to the applicant to acknowledge receipt of one or more Record LRPDUs.

¹²There is a fifth “Stop LRPDU” defined in 9.4.1, but this is merely a syntactic convention for padding an ECPDU to meet minimum frame length requirements on some physical media.

A Complete List LRPDU is sent periodically by the registrar to check whether the registrar database matches the applicant database.

See Clause 8 for descriptions of the state machines that transmit and receive these LRPDUs, and Clause 9 for their formats.

NOTE—The checksum on the data contained in Record LRPDUs is not computed for data integrity; LRP-DS runs over a transport protocol, either ECP or TCP, that supplies reasonable assurance that LRPDUs are delivered without being corrupted. The checksum is used to catch errors that can happen when one or both connected system are reinitialized, when two records can be transmitted with the same record identifier and sequence number, but with different data.

6.10.2 LRP-DS database storage

Figure 6-7 illustrates the entire LRP-DS data path, showing the transformations in the data as it is replicated. The procedure described here is somewhat more elaborate than an implementation might use, but it illustrates the underlying principles behind the design of LRP. The only parts of Figure 6-7 that constrain an implementation are the externally visible LRPDUs. Striped and blocks are used to represent pieces of information that are likely to have only a momentary existence. The numbered steps in Figure 6-7 are as follows:

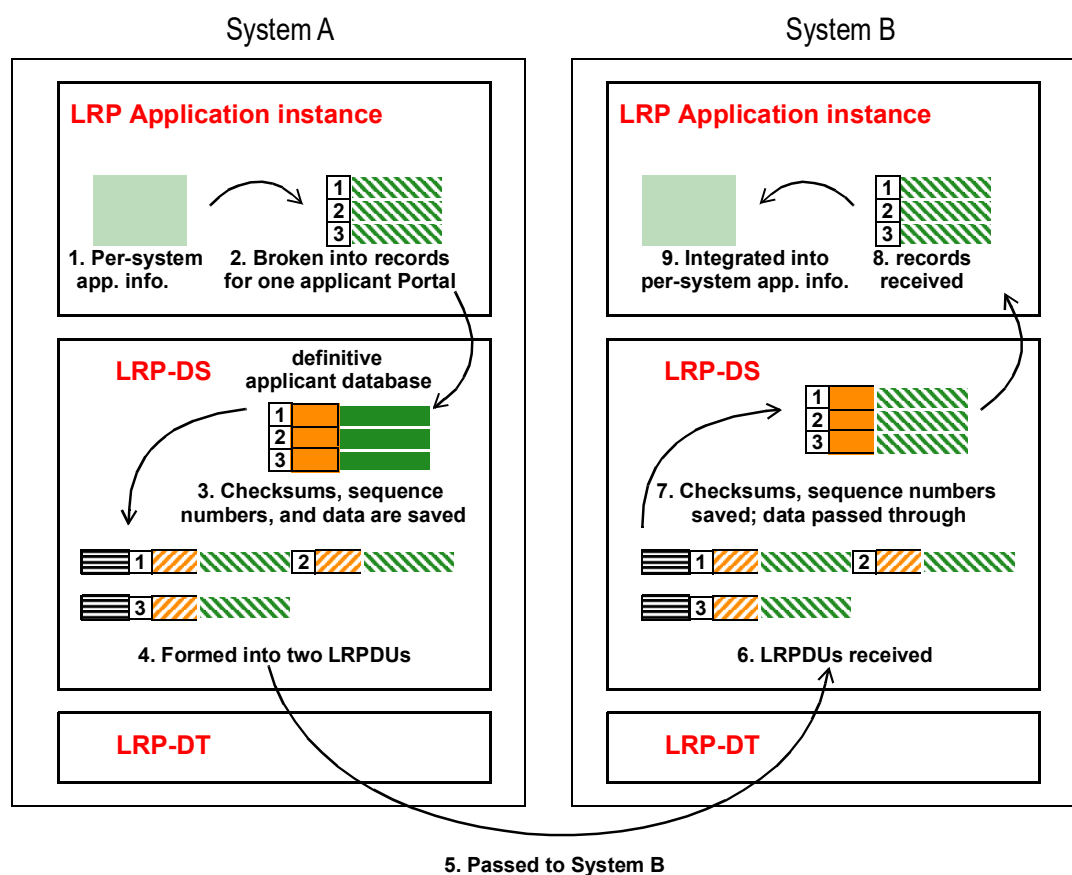


Figure 6-7—LRP-DS data path

- 1) We can posit “Per-system LRP application information” that contains information related to an application instance’s purpose.
- 2) We can further imagine that the application instance creates an image of an applicant database to be replicated to its partner on a particular port. From this image, the application instance creates a set of records, each with a record number, to be transmitted. The records may be a simple division of the image into sections, but see B.5 for another possibility.
- 3) The application instance uses the primitives in 10.3.1 to convey the records to the LRP-DS Portal, where the definitive copy of the applicant database resides. LRP-DS supplies a sequence number and checksum for each record.
- 4) As governed by the applicant state machines described in 8.3, LRP-DS collects the records that need to be transmitted into Record LRPDU’s (9.4.3), and passes them to LRP-DT.
- 5) LRP-DT moves the Record LRPDU’s from the applicant Portal to the registrar Portal, using acknowledgments, timeouts, and retries to deliver the data quickly and reliably.
- 6) Following the procedures described in 8.4, LRP-DS in the partner Portal receives the Record LRPDU’s from LRP-DT and splits out the various records.
- 7) LRP-DS responds with Partial List LRPDU’s as required, and saves the record numbers, sequence numbers, and checksums for each record. The actual data is not saved by LRP-DS, but is passed up to the application instance along with the record number and sequence number.
- 8) The application instance receives the numbered records (including deletions), and reconstructs the exact image of the applicant database (the registrar database), posited in item 2, above. This process may be as simple as concatenating the records in numerical order, but see B.5 for another possibility.
- 9) The application instance merges this registrar database into its per-system LRP application information, which may well trigger applicant-side operations on this or other Portals in the system.

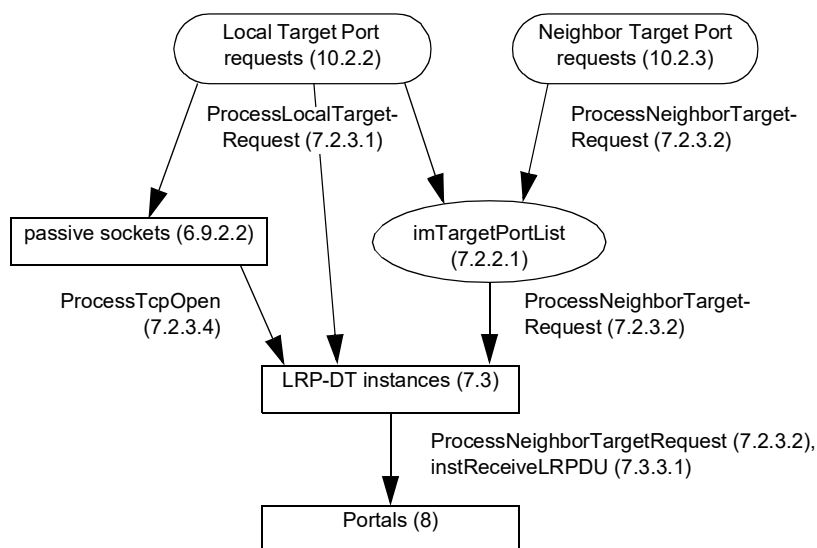
As a practical matter, a given implementation might merge the LRP application with LRP-DS, or might use a completely different set of primitives from those in Clause 10, but the specification of an LRP application is most easily described in terms of the Clause 10 primitives.

6.11 State machine creation

The LRP-DS and LRP-DT state machines can reside either in a Native system, with the target ports, or in a Proxy system, separate from the Controlled systems’ target ports (6.4). LRP-DS and LRP-DT state machines are created based on information supplied through the Local Target Port request and Neighbor Target Port request primitives (10.2.2, 10.2.3). The table `imTargetPortList` (7.2.2.1) stores this information. Changes to the `imTargetPortList` trigger the creation and destruction of LRP-DT state machines (7.3), which in turn lead to the creation and deletion of Portals (Clause 8). Figure 6-8 illustrates these relationships and shows the routines in Clause 7 that manage these tables and state machines.

Input to LRP from the application instance is required to actually create the LRP state machines and start their operation. As shown in Figure 6-8, LRP uses the `imTargetPortList` (7.2.2.1) to control the creation of ECP and TCP LRP-DT instances, as well as passive TCP sockets (6.9.2.2) to receive TCP connections from other systems. The receipt of TCP connection requests from other systems can create TCP LRP-DT instances. Also, ECP LRP-DT instances can be created directly by `ProcessLocalTargetRequest` (7.2.3.1) for the reception of Exploratory Hello LRPDU’s (6.6).

Each LRP-DT instance is one end of an LRP-DT connection to exactly one neighbor LRP-DT instance. One LRP-DT instance can be associated with one or more different LRP applications for one or more local target ports, each {target port, LRP application} pair constituting a Portal (Clause 8), which implements the LRP-DS sublayer.



A Portal and its state machines is created either with the creation of an LRP-DT instance, or after the receipt of the first Hello LRPDU (9.4.2), from a neighbor Portal.

A Portal transmits and receives LRPDUs through its associated LRP-DT instance. The Portals paired with a given LRP-DT instance share the transport connection to the neighbor LRP-DT instance.

6.12 Security considerations

LRP uses a transport layer, either ECP or TCP. ISO/IEC 10589:2002 IS-IS [B19] commonly uses security measures described in IETF RFC 5304 [B12]. However, since IS-IS does not use an underlying transport protocol, these measures are not of much use to LRP. Similarly, IETF RFC 7987 [B16] does not apply to LRP, because there is no Lifetime field to protect, although a Lifetime field could be useful to a particular LRP application.

The ECP protocol can use MACsec (IEEE Std 802.1AE™ [B4]) to provide a level of security for LRP. Because every data LRPDU carries a sequence number, replay of data or list LRPDUs is not a problem over either ECP or LRP. Replay of Hello LRPDUs could disrupt making a connection, but TCP prevents that problem. ECP has no replay protection for Hellos, but MACsec does prevent replays.

IPsec (IETF RFC 4301 [B11]) or Transport Layer Security (TLS, IETF RFC 8446 [B18]) can be used to further secure TCP connections. The use of these protocols is not specified in this standard.

7. LRP-DT Procedures

7.1 Introduction

This clause describes the following procedures and state machines:

- a) The mechanisms used to create and destroy LRP-DT instances are described in 7.2.
- b) The operation of an LRP-DT instance is described in 7.3.

7.2 LRP-DT instance maintenance

7.2.1 LRP-DT instance maintenance and Portal creation overview

LRP-DT instances are instantiations of the LRP-DT instance state machine (7.3.4), and exist only in a Native or a Proxy system. They are created and destroyed by the following methods:

- a) Local Target Port requests (10.2.2) issued by the application instance call `ProcessLocalTargetRequest` (7.2.3.1) to create and destroy descriptions of local target ports, that is, target ports residing in either that same Native system, or in a Controlled system controlled by the Proxy system in which the application instance resides.
- b) Similarly, Neighbor Target Port requests (10.2.3) issued by the application instance call `ProcessNeighborTargetRequest` (7.2.3.2) to create or destroy descriptions of neighbor target ports, each of which is attached to a specific local target port.
- c) If a local target port is described, but no neighbor target port is described, then
 - 1) If ECP is enabled (item b:3 in 10.2.2.1), then `ProcessLocalTargetRequest` creates an ECP LRP-DT instance, but no Portal.
 - 2) If a TCP address (and Port number) is supplied, then `ProcessLocalTargetRequest` creates a Passive TCP OPEN socket (6.9.2.2) if not already created, but no LRP-DT instance or Portal is created, yet.
- d) When a neighbor target port is described and attached to the local target port description, then:
 - 1) If the local and neighbor target ports have ECP addresses supplied, then `ProcessNeighborTargetRequest` creates a Portal and attaches it to the LRP-DT instance.
 - 2) If the local and neighbor target ports have TCP addresses (and TCP port number) supplied, then `ProcessNeighborTargetRequest` creates both a TCP LRP-DT instance and a Portal, which it attaches to the LRP-DT instance. It issues an Active TCP OPEN (6.9.2.1) on the LRP-DT instance, if one does not already exist.
- e) When an IEEE 1003.1 `accept()` call succeeds on a socket created in item c:2, above, `ProcessTcpOpen` (7.2.3.4) creates a new TCP LRP-DT instance, but not a Portal.
- f) When a Hello LRPDU (9.4.2) is received on an LRP-DT instance for an unknown neighbor target port, then First Hello indication (10.2.5) and Associate Portal request (10.2.4) are used to allow or disallow the Portal to associate with its neighbor.
- g) Local Target Port requests, Neighbor Target Port requests, and TCP connection failures can trigger the destruction of target port descriptions, which in turn, destroy the associated Portals and LRP-DT instances.
- h) Duplicated TCP connections can cause duplicated TCP LRP-DT instances. Subsequent operation of the Receive Hello state machine (8.2.5) can cause one of the duplicated LRP-DT instances and its TCP socket to be destroyed.

Only TCP LRP-DT instances need to establish connections with active or passive OPEN operations. ECP has no connection establishment operations.

The details of this process are described in the following subclauses.

7.2.2 Instance maintenance variables

7.2.2.1 imTargetPortList

An imTargetPortList resides in each Native or Proxy system. It maintains a list of all local and neighbor target ports of interest to the system. Entries in the imTargetPortList are created or destroyed by the ProcessLocalTargetRequest routine (7.2.3.1). Each entry in the list is associated with a single application instance and local target port, and thus has a unique value for the triple {imPplAppId, imPplMyChassisId, imPplMyPortId}.

In turn, each entry in the imTargetPortList has an attached list (imPplNeighborList, item g, below) of neighboring target ports. Entries in the imPplNeighborList are created or destroyed by the ProcessNeighborTargetRequest (7.2.3.2) routine. Each entry in the imPplNeighborList has, in conjunction with its parent imTargetPortList entry, a unique value for the 5-tuple {imPplAppId, imPplMyChassisId, imPplMyPortId, imPplNbrChassisId, imPplNbrPortId}.

The imTargetPortList is, in effect, a list of known target ports for which Portals are expected to be created. For each expected Portal, imTargetPortList has local and remote address information to be used for an LRP-DT instance to carry that Portal's LRPDU's. Note that this is not a list of all possible Portals, as Portals can also be created as a result of receiving Hello LRPDU's (item f in 7.2.1).

The relationship of the information in the imTargetPortList to that in the LRP ECP Discovery TLV and LRP TCP Discovery TLV (C.2.1, C.2.2) carried by IEEE 802.1AB LLDP is governed by the application instance via the primitives in Clause 10. See 6.5.

Each entry in the imTargetPortList contains the following items:

- a) **imPplAppId:** The AppId of an LRP application (9.2).
- b) **imPplMyChassisId:** The Chassis ID (9.4.2.6) of the Native or Controlled system's target port (8.5.2 of IEEE Std 802.1AB-2016).
- c) **imPplMyPortId:** The Port ID (9.4.2.7) of the Native or Controlled system's target port (8.5.3 of IEEE Std 802.1AB-2016).
- d) **imPplMyEcpEnable:** A Boolean value indicating whether this target port accepts LRPDU's transmitted over ECP.
- e) **imPplMyTcpInfo:** The LRP TCP IP address and TCP port number information for the Native or Controlled system's target port (C.2.2.6). This item can be empty.

NOTE 1—The procedures in 7.2.3.1 ensure that either imPplMyEcpEnable is TRUE, imPplMyTcpInfo is not empty, or that both conditions are met.

- f) **imPplHelloParms:** Hello parameters to be used by any generated Hello state machines (item c in 10.2.2.1).
- g) **imPplNeighborList:** A list, possible empty, of neighbor target ports. Each entry in the list contains:
 - 1) **imPplNbrChassisId:** The Chassis ID (9.4.2.8) the neighbor system's target port (8.5.2 of IEEE Std 802.1AB-2016).
 - 2) **imPplNbrPortId:** The Port ID (9.4.2.9) a neighbor system's target port (8.5.3 of IEEE Std 802.1AB-2016).

- 3) **imPplNbrEcpInfo:** The LRP ECP MAC address for a neighbor system's target port (C.2.1). This item can be empty.
- 4) **imPplExploreXmit:** A Boolean value indicating that the local Portal controlled by this imPplNeighborList entry can transmit Exploratory Hello LRPDU's (6.6).
- 5) **imPplExploreRecv:** A Boolean value indicating that the local Portal controlled by this imPplNeighborList entry is willing to receive Exploratory Hello LRPDU's (6.6).
- 6) **imPplNbrTcpInfo:** The LRP TCP IP address and TCP port number information for a neighbor system's target port (C.2.2.6). This item can be empty.

NOTE 2 —The procedures in 7.2.3.1 ensure that imPplNbrEcpInfo and imPplNbrTcpInfo are never both empty.

- 7) **imPplActiveOpen:** A Boolean value indicating that this system needs to issue an active TCP connection OPEN operation (6.9.2.1) for this target port pair.
- 8) **imPplPortalRef:** A reference (1.5) to the Portal associated with this neighbor target port, or null, if none.

A system shall not allow the creation of an imTargetPortList such that any of the following are true:

- h) Two or more different imPplNeighborList entries have the same values of the 5-tuple {imPplAppId, imPplMyChassisId, imPplMyPortId, imPplNbrChassisId, imPplNbrPortId}.
- i) In one imPplNeighborList entry, imPplNbrEcpInfo is null and either imPplExploreXmit or imPplExploreRecv is TRUE.

7.2.3 Instance maintenance routines

7.2.3.1 ProcessLocalTargetRequest

In a Native or Proxy system, a Local Target Port request (10.2.2) results in a call to ProcessLocalTargetRequest to create or destroy an entry in the imTargetPortList (7.2.2.1), the list of {LRP application, target port} pairs in the system. This routine takes the parameters given in the Local Target Port request as inputs. ProcessLocalTargetRequest then modifies the imTargetPortList as follows:

- a) If the parameters of the Local Target Port request are invalid (e.g., specify a non-existent target port), theProcessLocalTargetRequest terminates and returns an error code for the Local Target Port request (item c:1 in 10.2.2.2).
- b) If no entry exists in the imTargetPortList with imPplAppId, imPplMyChassisId, and imPplMyPortId values matching the AppId, Chassis ID, and Port ID values specified in the Local Target Port request (item a:1, item b:1, and item b:2 in 10.2.2.1), then:
 - 1) If ECP is disabled and no IP address information is supplied in the Local Target Port request, the request is ignored and ProcessLocalTargetRequest terminates.
 - 2) Otherwise (ECP enabled and/or IP address is supplied), ProcessLocalTargetRequest creates a new entry in the imTargetPortList, and fills the imPplMyEcpEnable, imPplMyTcpInfo, and imPplHelloParms values in the new imTargetPortList entry from the information specified in the Local Target Port request (item b:3, item b:4, item c:2 in 10.2.2.1). The imPplNeighborList item in the new entry is empty. Processing continues (item d, below).
- c) Otherwise (a matching entry is found):
 - 1) If ECP is disabled and no IP address information is supplied in the Local Target Port request, the existing imTargetPortList entry is deleted. Processing continues (item d, below).
 - 2) Otherwise (ECP enabled and/or IP address is supplied), If the imPplMyEcpEnable, imPplMyTcpInfo, and imPplHelloParms values in the imTargetPortList entry match the information specified in the Local Target Port request (item b:3, item b:4, item c:2 in 10.2.2.1),

request is ignored and ProcessLocalTargetRequest terminates. If they do not match, an error is reported (item c:1 in 10.2.2.2) and ProcessLocalTargetRequest terminates.

- d) If an entry in the imTargetPortList is created with a value for an IPv4 and/or an IPv6 address in imPplMyTcpInfo (item e in 7.2.2.1) that was not previously in the imTargetPortList, then ProcessLocalTargetRequest creates a new socket and issues an IEEE 1003.1 `listen()` call to initiate a passive TCP OPEN for each new IP address (6.9.2.2).
- e) If item c:1, above, resulted in the deletion of an imTargetPortList entry, then ProcessLocalTargetRequest performs calls ProcessNeighborTargetRequest (7.2.3.2) on each entry in the imPplNeighborList (item g in 7.2.2.1) of the imTargetPortList entry to delete the neighbors and their Portals. These actions can cause the destruction of LRP-DT instances.
- f) If any deleted imTargetPortList entry was the only remaining entry in the list to specify a particular IP address, then the socket on which the IEEE 1003.1 `listen()` function was issued for passive TCP OPENs for that IP address is destroyed with the IEEE 1003.1 `shutdown()` call.

7.2.3.2 ProcessNeighborTargetRequest

Whenever a Neighbor Target Port request (10.2.3) is made in a Native or Proxy system, the system calls ProcessNeighborTargetRequest to create or destroy an LRP-DT instance and Portal for a specific neighbor target port. ProcessNeighborTargetRequest is also called by the Receive Hello state machine (8.2.5) when a Portal Association is approved by the application instance using the Associate Portal request (10.2.4). ProcessNeighborTargetRequest takes the parameters given in the Neighbor Target Port request as inputs in the former case, and takes the fields in the received Hello LRPDU (9.4.2) in the latter case. The routine then modifies the imPplNeighborList (item g in 7.2.2.1) in the imTargetPortList entry specified in the Neighbor Target Port request (7.2.2.1, item c:1 in 10.2.3.1) as follows:

- a) If the parameters of the Neighbor Target Port request are invalid (e.g., specify a non-existent imTargetPortList entry), ProcessNeighborTargetRequest terminates and returns an error code for the Neighbor Target Port request (item b:1 in 10.2.3.2).
- b) If no entry exists in the selected imPplNeighborList with imPplAppId, imPplNbrChassisId, and imPplNbrPortId values (item a, item h:1, item g:2 in 7.2.2.1) matching the AppId, Chassis ID, and Port ID values specified in the Neighbor Target Port request (item e:1 in 10.2.2.1, item b:1 and item b:2 in 10.2.3.1), then:
 - 1) If no ECP address or IP address information is supplied in the Neighbor Target Port request, the request is ignored and ProcessNeighborTargetRequest terminates.
 - 2) Otherwise (ECP address and/or IP is supplied), ProcessNeighborTargetRequest creates a new entry in the imPplNeighborList, and fills the imPplNbrEcpInfo, imPplExploreXmit, imPplExploreRecv and imPplNbrTcpInfo values in the new imPplNeighborList entry from the ECP address and IP address information specified in the Neighbor Target Port request (item b:3, item b:4 in 10.2.3.1). ProcessNeighborTargetRequest sets imPplActiveOpen to FALSE and imPplPortalRef to null (item h:7, item g:8 in 7.2.2.1). Processing continues (item d, below).
- c) Otherwise (a matching entry is found):
 - 1) If no ECP address or IP address information is supplied in the Neighbor Target Port request, then `instDestroyPortal` (7.3.3.5) is called to delete the portal referenced in the imPplPortalRef (item g:8 in 7.2.2.1) of the deleted entry, if any. The imPplNeighborList entry is then deleted. ProcessNeighborTargetRequest terminates.
 - 2) Otherwise (ECP address and/or IP is supplied), if the imPplNbrEcpInfo and imPplNbrTcpInfo values in the imPplNeighborList entry do not match the ECP address and IP address information specified in the Neighbor Target Port request (item b:3, item b:4 in 10.2.3.1), an error is reported (item b:1 in 10.2.3.2) and ProcessNeighborTargetRequest terminates.
 - 3) Otherwise (imPplNbrEcpInfo and imPplNbrTcpInfo values in the imPplNeighborList entry do match the ECP address and IP address values specified in the Neighbor Target Port request),

then `ProcessNeighborTargetRequest` examines all LRP-DT instances. If a TCP LRP-DT instance is found that lists the same `imTargetPortList` entry in its `instTargetPortList` (7.3.2.8) that was given in the Neighbor Target Port request, and has the same IP addresses as the one returned by `PickIpAddresses`, then the request is ignored and `ProcessNeighborTargetRequest` terminates. Otherwise, the `imPplNeighborList` entry is selected for further processing.

NOTE—Because LRP-DT reuses TCP connections in order to conserve resources (item d:1, below), a connection that would have been selected as an active TCP connection can make use of an existing LRP-DT instance's passive TCP connection. If that connection is lost, the application instance can issue a new Neighbor Target Port request (10.2.3) for an already-extant `imPplNeighborList`, whereupon `ProcessNeighborTargetRequest` will initiate a new (active) TCP connection, if necessary. This is the reason that this last bullet (item c:3) can cause a new LRP-DT instance to be created for an already-extant `imPplNeighborList` entry that does not have one.

- d) At this point, a new `imPplNeighborList` entry has been created, or an existing entry has been selected for which there is no LRP-DT instance. `PickIpAddresses` (7.2.3.3) is called with `imPplMyTcpInfo` and `imPplNbrTcpInfo` (item e and item g:6 in 7.2.2.1). If `PickIpAddresses` indicates success, and also indicates that an active TCP connection is to be made, then:
 - 1) `ProcessNeighborTargetRequest` examines all LRP-DT instances. If an LRP-DT instance is found that lists the same `imTargetPortList` entry in its `instTargetPortList` (7.3.2.8) that was given in the Neighbor Target Port request, and has the same IP addresses as the one returned by `PickIpAddresses`, then that LRP-DT instance is selected for further processing by `ProcessNeighborTargetRequest`.
 - 2) Otherwise, (no matching LRP-DT instance was found), then a new LRP-DT instance is created by calling `instCreateInstance` (7.3.3.2) with the `imPplNeighborList` entry information and the information returned by `PickIpAddresses`. This LRP-DT instance is selected for further processing by `ProcessNeighborTargetRequest`. `instActiveTcp` (7.3.2.1) is set to TRUE. Normal operation of the newly-created LRP-DT instance state machine (7.3.4) will trigger an Active TCP OPEN (6.9.2.1).
- e) If `PickIpAddresses` succeeded, but indicated that no active TCP is to be made, then `ProcessNeighborTargetRequest` terminates processing successfully.
- f) If `PickIpAddresses` failed, then the `imPplMyEcpEnable` and `imPplNbrEcpInfo` (item d and item g:3 in 7.2.2.1) are examined. If neither is empty, then an ECP LRP-DT instance is created, by calling `instCreateInstance` (7.3.3.2), and selected for further processing by `ProcessNeighborTargetRequest`. If both are empty, a failure indication is returned by `ProcessNeighborTargetRequest` (item b:1 in 10.2.3.2).
- g) Having selected or created an ECP or active TCP LRP-DT instance, `ProcessNeighborTargetRequest` calls `instCreatePortal` (7.3.3.4) to create a Portal, if one does not already exist. `pamNeedHello` is set TRUE for a TCP LRP-DT instance or an ECP LRP-DT instance whose `imPplExploreXmit` flag is TRUE, else sets `pamNeedHello` to FALSE. `ProcessNeighborTargetRequest` also adds a reference to the Portal to `imPplPortalRef` (item g:8 in 7.2.2.1) and to the LRP-DT instance's `instPortalList` (7.3.2.9).

7.2.3.3 PickIpAddresses

This routine takes as inputs two Application descriptor fields (C.2.2.6): `imPplMyTcpInfo` (item e in 7.2.2.1), representing the local (Native or Proxy system's) IP address information, and `imPplNbrTcpInfo` (item g:6 in 7.2.2.1), representing the remote system's IP address information. `PickIpAddresses` returns as output either a failure indication, or the following information:

- a) A "this end" IP address,
- b) An "other end" IP address (or "unknown"),
- c) An indication of whether the addresses (a and b) are IPv4 or IPv6,
- d) A Boolean value indicating that this system should perform an active TCP OPEN (6.9.2.1) as part of creating an LRP-DT instance.

PickIpAddresses operates as follows:

- e) If both Application descriptors' Address info fields (C.2.2.6.2) indicate IPv6 addresses (addrIPv6), then the IPv6 addresses are returned.
- f) Otherwise (one or the other of the address fields did not indicate an IPv6 address), if both Application descriptors' Address info fields (C.2.2.6.2) indicate IPv4 addresses (addrIPv4), then the IPv4 addresses are returned.
- g) Otherwise (one Application descriptor indicates IPv4 only, and the other IPv6 only), a failure indication is output.

The Address info fields (C.2.2.6.2) of inputs are then examined, and a return value for the active TCP OPEN choice is taken from Table 7-1. (See B.7 for a discussion of the need for Table 7-1.)

Table 7-1—Active TCP OPEN required

	Neighbor's Active/Passive preference		
My Active/Passive preference	apNoPref	apActive	apPassive
apNoPref	TRUE	FALSE	TRUE
apActive	TRUE	TRUE	TRUE
apPassive	FALSE	FALSE	TRUE

The principles driving this table are that:

- Both sides always make a passive TCP OPEN (6.9.2.2).
- If the preferences gives a clear choice as to which makes an active OPEN (6.9.2.2), that choice is respected.
- If neither party has a preference, or if their preferences conflict, both systems make an active OPEN.

7.2.3.4 ProcessTcpOpen

ProcessTcpOpen runs continuously in a Native or Proxy system supporting the LRP-DT TCP mechanism (6.9.2). It issues an IEEE 1003.1 `accept()` on every passive OPEN socket created by ProcessLocalTargetRequest (7.2.3.1). Whenever an `accept()` call succeeds, indicating that a remote system has issued an active OPEN (6.9.2.2), ProcessTcpOpen creates a TCP LRP-DT instance (7.3) with `instActiveTcp` (7.3.2.1) set to FALSE and `instNeighborAddress` and `instNeighborPortNumber` (7.3.2.4, 7.3.2.5) set to the remote IP address obtained from the `accept()` call. The new LRP-DT instance's `instPortalList` (7.3.2.9) is empty; no Portal can be created until a Hello LRPDU is received (7.3.3.1).

NOTE—The operation of IEEE 1003.1 sockets ensures that the newly created LRP-DT instance will have a different pair of IP addresses and port numbers from any other TCP LRP-DT instance.

7.3 LRP-DT instance

7.3.1 LRP-DT instance overview

An LRP-DT instance is an instantiation of the LRP-DT instance state machine (7.3.4). Its purposes are to:

- a) Establish a connection to another LRP-DT instance via TCP. (No connection is needed for ECP.)
- b) Create and/or destroy Portals' state machines (8.2, 8.3, 8.4).
- c) Receive LRPDU and assign them to a Portal for processing.

An LRP-DT instance serves some number of target ports (7.3.2.8). An ECP LRP-DT instance runs in a Native system only, and sends and receives all of its LRPDU through the single target port that it serves. A TCP LRP-DT instance runs in a Native or Proxy system, can serve any number of local target ports, and can send or receive its LRPDU through any port, or different ports at different times.

7.3.2 LRP-DT instance variables

7.3.2.1 instActiveTcp

A Boolean value that is TRUE if and only if instMyAddress (7.3.2.2) and instNeighborAddress (7.3.2.4) are TCP addresses and this LRP-DT instance is using the active (6.9.2.1), not the passive (6.9.2.2), form of TCP OPEN.

7.3.2.2 instMyAddress

The address of the local system for this LRP-DT instance; the address used as a destination address by the neighbor LRP-DT instance. The address includes a type (MAC, IPv4, or IPv6) and an address of that type.

7.3.2.3 instMyPortNumber

The local port number for this TCP connection, or 0, if this connection uses ECP, instead of TCP.

7.3.2.4 instNeighborAddress

The address of the neighbor LRP-DT instance; the address used as a destination address by this LRP-DT instance. The address includes a type (MAC, IPv4, or IPv6) and an address of that type.

7.3.2.5 instNeighborPortNumber

The remote port number for this TCP connection, or 0, if this connection uses ECP, instead of TCP.

7.3.2.6 instReconnectTimer

An integer valued timer that decrements once every second, if and only if its value is non-zero.

7.3.2.7 instReconnectReset

An integer number of seconds to wait before trying to remake the next failed TCP connection.

7.3.2.8 instTargetPortList

A list of references to the entries in the imTargetPortList (7.2.2.1) attached to this LRP-DT instance. This is a list of target ports in the Native or Controlled system served by this LRP-DT instance.

7.3.2.9 instPortalList

A list of references to the Portals attached to this LRP-DT instance.

7.3.2.10 instKillInstance

A Boolean signal indicating that the LRP-DT instance state machine is to cease operation.

7.3.3 LRP-DT instance routines

7.3.3.1 instReceiveLRPDU

This routine is called whenever an LRPDU is received on an LRP-DT instance. The LRPDU is processed according to the following:

- a) Stop LRPDU (9.4.1) are discarded without further processing.
- b) If the LRPDU type field (9.3.2) of the LRPDU is not one of the values typeHelloLRPDU, typeRecordLRPDU, typePartialListLRPDU, or typeCompleteListLRPDU from Table 9-3, the LRPDU is counted in instDiscardedLRPDUs (11.4.1) and discarded without further processing.
- c) If a Partial List LRPDU (9.4.4), Complete List LRPDU (9.4.5), or Record LRPDU (9.4.3) is received, instReceiveLRPDU searches the list of Portals in the LRP-DT instance's instPortalList (7.3.2.9) for a Portal whose pamNeighborPortalNumberValid (8.2.2.6) is TRUE, and whose pamNeighborPortalNumber (8.2.2.7) matches the My Portal Number field (9.4.2.4) of the received LRPDU. If none is found, the LRPDU is counted in instDiscardedLRPDUs (11.4.1) and discarded without further processing. If found, then the LRPDU is presented to the Portal using actReceivePartialList (8.3.3.3), actReceiveCompleteList (8.3.3.4), or regReceiveWriteRecord (8.4.4.1), according to the LRPDU type, and instReceiveLRPDU terminates. This standard does not specify the action to be taken by instReceiveLRPDU if more than one of its attached Portals has the same value for pamNeighborPortalNumber.
- d) Otherwise (a Hello LRPDU, 9.4.2, is received), instReceiveLRPDU checks to see whether or not both of the Neighbor Chassis ID TLV and Neighbor Port ID TLV are present in Hello LRPDU, and proceeds as follows:
 - 1) If both TLVs are missing, the LRP-DT instance is an ECP LRP-DT instance, there is only one imTargetPortList entry attached to the LRP-DT instance (instTargetPortList, 7.3.2.8), and that entry's imPplNeighborList contains an entry with the imPplExploreRecv flag = TRUE (item g:5 in 7.2.2.1), then an exploratory Hello LRPDU (6.6) has been received. Processing of the Hello LRPDU proceeds with item e, below.
 - 2) Otherwise (if any of the conditions of item d:1 are not met), instReceiveLRPDU scans just those imTargetPortList entries listed in the LRP-DT instance's instTargetPortList (7.3.2.8) for an entry whose imPplAppId, imPplMyChassisId, and imPplMyPortId fields (item a, item b, and item c in 7.2.2.1) match the appId field, Neighbor Chassis ID TLV, and Neighbor Port ID TLV of the received Hello LRPDU (9.4.2.1, 9.4.2.8, 9.4.2.9). If a matching entry is found, then processing proceeds with item e, below.
 - 3) Otherwise (no match is found), this LRP-DT instance does not support the target port named by the Hello LRPDU; the LRPDU is counted in instDiscardedLRPDUs (11.4.1) and discarded, and no further processing takes place.
- e) At this point, we know that the LRP-DT instance serves the local target port named (except item d:1, above) in the Hello LRPDU. A Portal to process this Hello LRPDU is either found or created, in the following manner:
 - 1) The selected imTargetPortList's imPplNeighborList (item g in 7.2.2.1) is searched for an entry whose imPplNbrChassisId and imPplNbrPortId values (item h:1, item g:2 in 7.2.2.1) match the My Chassis ID TLV and My Port ID TLV, respectively, from the Hello LRPDU. In this search, an imPplNeighborList entry with the imPplExploreRecv flag set and null values for imPplNbrChassisId and imPplNbrPortId is deemed to match any Hello LRPDU. This enables the receipt of an exploratory Hello LRPDU (6.6). If a match is found, then the Portal named in