BS EN 62439-7:2012

Incorporating corrigendum May 2015

BSI Standards Publication

Industrial communication networks — High availability automation networks

Part 7: Ring-based Redundancy Protocol (RRP)

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 62439-7:2012. It is identical to IEC 62439-7:2011, incorporating corrigendum May 2015.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications: process measurement and control, including fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Published by BSI Standards Limited 2015

ISBN 978 0 580 90610 7

ICS 25.040; 35.040

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 April 2012.

Amendments/corrigenda issued since publication

Date	Text affected
31 July 2015	Implementation of IEC corrigendum May 2015: Clause 10 updated

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 62439-7

April 2012

ICS 25.040; 35.040

English version

Industrial communication networks - High availability automation networks - Part 7: Ring-based Redundancy Protocol (RRP)

(IEC 62439-7:2011)

Réseaux de communication industriels - Réseau de haute disponibilité pour l'automation - Partie 7: Protocole de redondance pour réseau en anneau (RRP) (CEI 62439-7:2011)

Industrielle Kommunikationsnetze - Hochverfügbare Automatisierungsnetze - Teil 7: Protokoll für ringbasierte Redundanz (RRP) (IEC 62439-7:2011)

This European Standard was approved by CENELEC on 2012-01-20. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization

Comité Européen de Normalisation Electrotechnique

Europäisches Komitee für Elektrotechnische Normung

Management Centre: Avenue Marnix 17, B - 1000 Brussels

Foreword

The text of document 65C/668/FDIS, future edition 1 of IEC 62439-7, prepared by SC 65C, "Industrial networks", of IEC TC 65, "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62439-7:2012.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2012-10-20
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2015-01-20

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 62439-7:2011 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following note has to be added for the standard indicated:

IEC 61158 series NOTE Harmonized in EN 61158 series.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

Publication	Year	Title	EN/HD	Year
IEC 60050-191	-	International Electrotechnical Vocabulary (IEV) - Chapter 191: Dependability and quality of service	-	-
IEC 62439-1	2010	Industrial communication networks - High availability automation networks - Part 1: General concepts and calculation methods	EN 62439-1	2010
ISO/IEC 8802-3	2000	Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications	-	-

CONTENTS

INTRODUCTION

The IEC 62439 series specifies relevant principles for high availability networks that meet the requirements for industrial automation networks.

In the fault-free state of the network, the protocols of the IEC 62439 series provide ISO/IEC 8802-3:2000 (IEEE 802.3) with compatible, reliable data communications, and preserve determinism in real-time data communications. In cases of fault, removal, and insertion of a component, they provide deterministic recovery times.

These protocols retain fully the Ethernet communication capabilities typically used in the office world, to ensure that software that relies on these protocols will remain applicable.

The market is in need of several network solutions, each with different performance characteristics and functional capabilities, meeting diverse application requirements. These solutions support different redundancy topologies and mechanisms, which are introduced in IEC 62439-1 and specified in the companion International Standards. IEC 62439-1 also distinguishes between these different solutions, providing guidance for the user.

The IEC 62439 series follows the general structure and terms of IEC 61158 series.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning IEC 61158-4-21 given in Clause 4 and Clause 5.

Patent Number KR 0789444 "COMMUNICATION PACKET PROCESSING APPARATUS AND METHOD FOR RING TOPOLOGY ETHERNET NETWORK CAPABLE OF PREVENTING PERMANENT PACKET LOOPING," owned by LS INDUSTRIAL SYSTEMS CO., LTD., Anyang, Korea

Patent Number KR 0732510 "NETWORK SYSTEM" owned by LS INDUSTRIAL SYSTEMS CO., LTD., Anyang, Korea

Patent Number KR 0870670 "Method For Determining a Ring Manager Node", owned by LS INDUSTRIAL SYSTEMS CO., LTD., Anyang, Korea

IEC takes no position concerning the evidence, validity and scope of these patent rights.

The holder of these patent rights has assured the IEC that he/she is willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of these patent rights is registered with IEC. Information may be obtained from:

LSIS Co Ltd
LS Tower
1026-6, Hogye-Dong
Dongan-Gu
Anyang, Gyeonggi-Do, 431-848
South Korea
Phone +82 2 2034 4917
Fax +82 2 2034 4648

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line data bases of patents relevant to their standards. Users are encouraged to consult the data bases for the most up to date information concerning patents.

INDUSTRIAL COMMUNICATION NETWORKS – HIGH AVAILABILITY AUTOMATION NETWORKS –

Part 7: Ring-based Redundancy Protocol (RRP)

1 Scope

The IEC 62439 series of standards is applicable to high-availability automation networks based on the ISO/IEC 8802-3:2000 (Ethernet) technology.

This part of the IEC 62439 series specifies a redundancy protocol that is based on a ring topology, in which the redundancy protocol is executed at the end nodes, as opposed to being built into the switches. Each node detects link failure and link establishment using media-sensing technologies, and shares the link information with the other nodes, to guarantee fast connectivity recovery times. The nodes have equal RRP network management functions.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60050-191, *International Electrotechnical Vocabulary – Chapter 191 : Dependability and quality of service*

IEC 62439-1:2010, *Industrial communication networks – High availability automation networks – Part 1: General concepts and calculation methods*

ISO/IEC 8802-3:2000, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

3 Terms, definitions, abbreviations, acronyms, and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 60050-191 as well as in IEC 62439-1, and the following apply.

3.1.1
R-port
port in a communication device that is part of a line or ring structure

3.1.2
device address
2 octet address that designates the device associated with a single device on a specific local link

3.1.3
Gateway Device
GWD
RRP device that has more than 3 Ethernet ports on it. At least 2 ports have to support RRP protocol

3.1.4
Normal Device
ND
normal RRP device which has two RRP ports on it

3.1.5
Unique Identification
UID
Unique 8 octet identification used to identify a RRP device within a network segment. UID is combines a 2 octet device address and a 6 octet MAC address, so that it has a unique value in a network

3.2 Abbreviations and acronyms

For the purposes of this document, the abbreviations and acronyms given in IEC 62439-1 as well as the following, apply.

ASE	Application Service Element
DLE	Data Link layer Entity
FC	Frame Control
FCS	Frame Check Sequence
GD	General Device
GWD	Gateway Device
LNM	Line Network Manager
MAC	Media Access Control
MIB	Management Information Base
NCM	Network Control Message
NCMT	Network Control Message Type
ND	Normal Device
NMIB	Network Management Information Base
PHY	Physical Interface Transceiver
PO	Power On
PRI	Priority
RES	Reserved
RNM	Ring Network Manager
RRP	Ring based Redundancy Protocol
SA	Stand Alone
ToS	Type of Service
VoE	Validation of Extension code

3.3 Conventions

This part of the IEC 62439 series follows the conventions defined in IEC 62439-1.

4 RRP overview

4.1 General

The RRP specifies a recovery protocol, based on a ring topology. All links in an RRP network shall be full duplex through the use of an internal hardware Ethernet switch. Thus, RRP provides a collision-free transmission mechanism between two nodes. Every RRP device detects link failure and link establishment using the rules specified in ISO/IEC 8802-3:2000 and shares this information with other RRP devices so that fast connectivity recovery time is also guaranteed in the ring network.

A RRP device is a dual-port switching device that receives and transmits standard ISO/IEC 8802-3:2000 Ethernet frames. It is intelligent and can control directional frame forwarding between its dual ports according to the network status and device status. RRP uses a special network management scheme specified in this standard. RRP also uses a network control based on device address and MAC address, and thus general bridge hub or switch might not be suitable for RRP network. However, when connecting a general Ethernet device to RRP network, Gateway Device (GWD) should be used.

4.2 Frame forwarding and receiving control

4.2.1 General

RRP provides a collision-free transmission mechanism with an internal full-duplex hardware switch with switching queue and dual MACs in a device. The switching priority method between Tx and Forwarding can be Round-Robin, Tx-First or Forwarding-First scheme. However RRP does not specify the switching method.

Thus, a RRP device transmits frames without the restriction of medium access, as soon as they appear in the transmit queue for each MAC. Figure 1 shows the forwarding and receiving control of the RRP device.

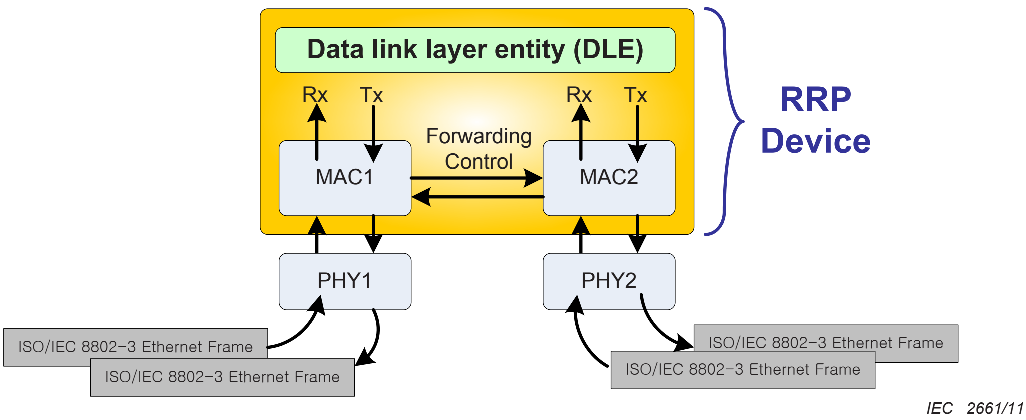

Figure 1 – Forwarding and receiving Ethernet frames

4.2.2 Normal Device (ND) and Gateway Device (GWD)

RRP is operated in a dual-port ring topology. A general Ethernet device can send standard Ethernet frames through RRP ring network with GWD. Multi-ring network can also be established using GWD.

GWD is responsible for switching Ethernet frames between RRP network and external Ethernet networks through application layer using a dynamic table. The dynamic table maps addresses to external Ethernet ports automatically. The dynamic table is automatically made by learning frame movements in the network. The GWD inspects both the destination and the source addresses. The destination address is used for the forwarding decision; the source address is used for adding entries to the table and for updating purposes. When an Ethernet frame is received at the media access control (MAC) layer through the physical interface transceiver (PHY), a GWD handles the received frame by taking one of the following actions, depending on the destination MAC address and the source MAC addresses in the received frame:

- for a broadcast or multicast frame, accept and deliver the frame to the data link layer entity (DLE), and forward the frame to the other RRP port and external Ethernet ports;
- for a frame designated for the device itself, accept and deliver the frame to the DLE without forwarding;
- for a frame designated for another device, accept the frame to its application layer and inspect both the destination and the source addresses. When the destination address of the frame is in the dynamic table, the GWD delivers the frame to the corresponding port in the dynamic table without forwarding to other ports. Otherwise, the GWD delivers the frame to all other ports. The GWD adds this entry to the dynamic table with source MAC address and port number information.

NOTE Dynamic table entries are automatically removed after the Ageing Time which is specified in IEEE 802.1D.

Figure 2 shows different structures of ND and GWD. In GWD, external Ethernet connection is connected to RRP ring network through MAC_E and PHY_E.

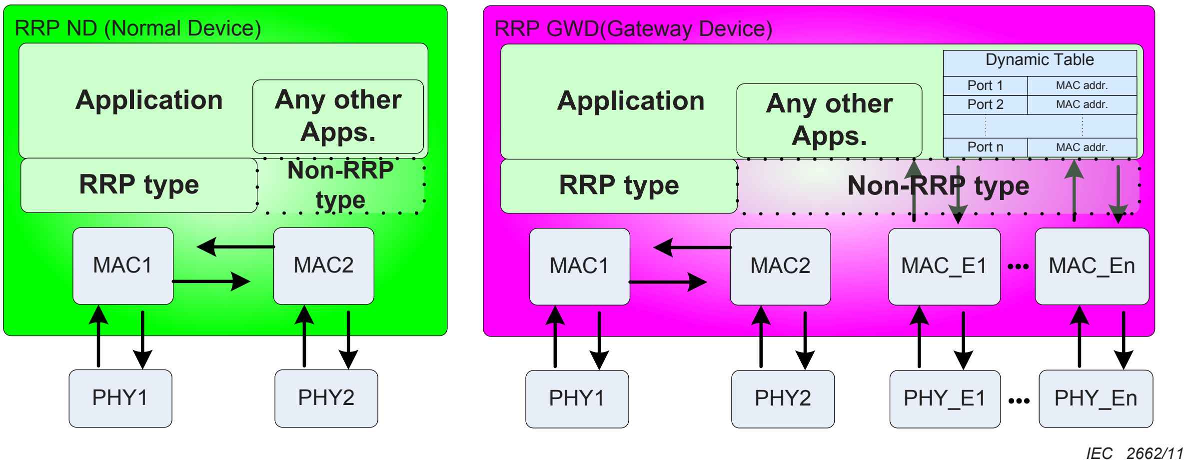

Figure 2 – Structures of ND and GWD

4.2.3 Behaviours of the General Device (GD)

When an Ethernet frame is received at the MAC layer through the PHY, a RRP general device other than the ring network manager (RNM) or the line network manager (LNM), handles the received frame by taking one of the following actions, depending on the destination MAC address and the device address in the received frame:

- for a broadcast or multicast frame, accept and deliver the frame to the DLE, and forward the frame to the other port;
- for a frame designated for the device itself, accept and deliver the frame to the DLE without forwarding;
- for a frame designated for another device, do not accept the received frame, but forward the frame to the other port.

This frame forwarding procedure is processed by the internal hardware switch, so that it has little impact on the performance of the RRP protocol.

4.2.4 Behaviours of the Line Network Manager (LNM)

As shown in Figure 3, the LNM disables the frame forward functions in both directions, so that frames are not forwarded to another port. In RRP networks, a LNM is automatically configured. When a device senses that only one port is connected, the device takes this to indicate that it is at the end of the line network. The LNM also becomes a control point of the hop count to other devices in a line network.

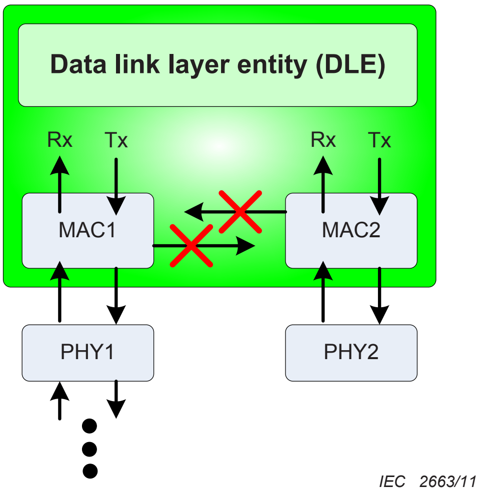

Figure 3 – LNM forwarding control

4.2.5 Behaviours of the Ring Network Managers (RNMs)

A frame in a ring network can be continuously circulated when the designated device is not found or when the frame is broadcast on the network. In a RRP ring network, two RNMs are automatically selected, and each RNM enables only one directional frame forward function to prevent infinite frame circulation, as shown in Figure 4.

The dual RNM structure is used to avoid message duplication. A primary RNM (RNMP) is selected with the highest UID device first, and then one of its neighbouring nodes is selected as a secondary RNM (RNMS). The RNMP and RNMS send Network Control Message Type (NCMT) messages to each other, to monitor network integrity.

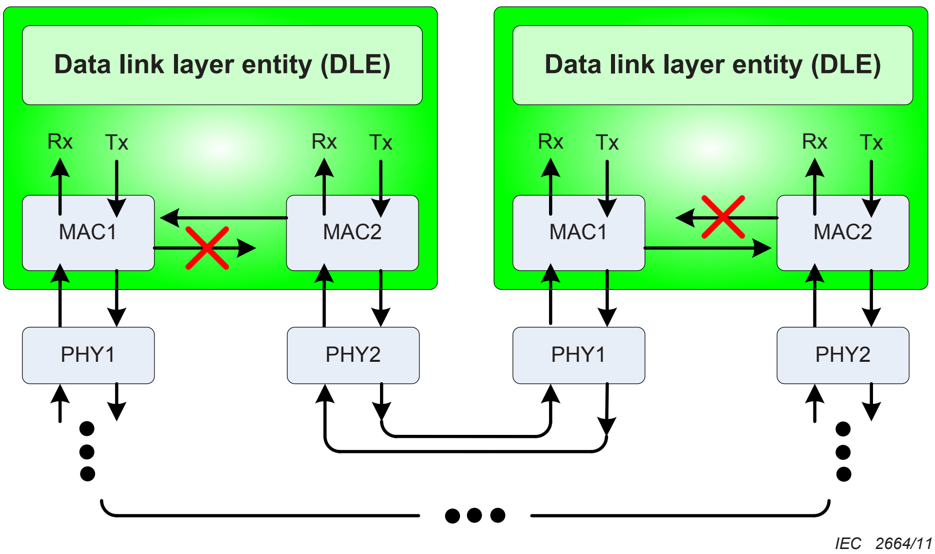

Figure 4 – RNM forwarding control

4.3 Link status monitoring

The RRP manages the network dynamically. When a link between two devices is established or released, it is automatically detected in the physical layer, as specified in ISO/IEC 8802-3:2000, Clause 24. This link status information is distributed and shared with every device on the network using NCMT messages, so that the network topology can be managed dynamically. The link status information is either “PHY_LINK_UP” or “PHY_LINK_DOWN” and the link status detection process is initiated by the sublayer of PHY service. A status of “PHY_LINK_UP” means that a RRP communication link is connected between two devices and it is possible to send frames through the link. A status of “PHY_LINK_DOWN” means that a RRP communication link is not established through an Ethernet MAC port and it is not possible to send frames through the port. By sharing all the link information on the network, all RRP devices on the network can determine the online network connectivity status. Figure 5 shows the intrinsic link status monitoring procedure of the RRP device.

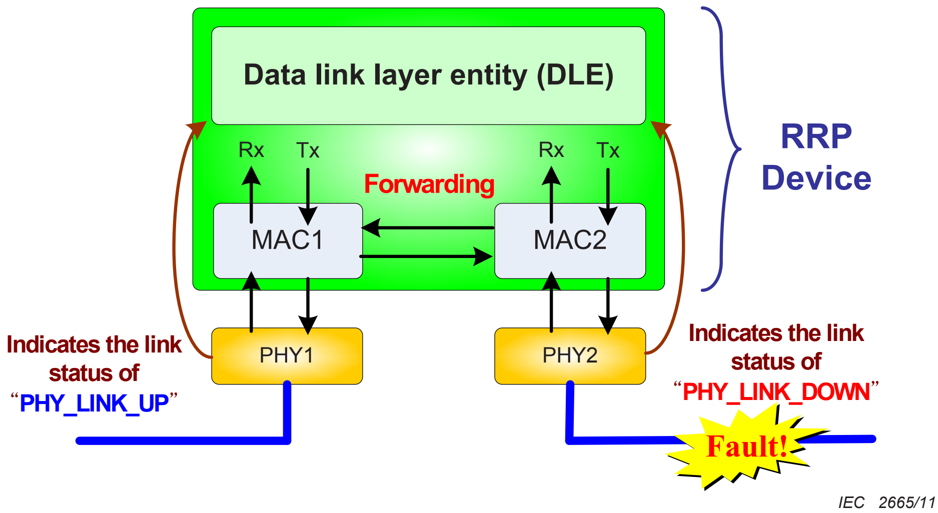

Figure 5 – Link status information

4.4 Error detection

A RRP device examines both frame validation and physical link status. Frame validation is examined using the frame check sequence (FCS) of ISO/IEC 8802-3:2000, Clause 3. The physical link status can be validated by a PHY link monitoring function. RRP uses a service of PHY sublayer to monitor link status.

4.5 Plug and play

When a new device joins an existing network, the new link information is broadcast via a NCMT message to every device on the network. The new device also collects existing link information from each device so that it can communicate to the other nodes on the network without manual configuration.

4.6 Network management information base (NMIB) management

A RRP device automatically manages network information and a path table. Network information and the path table are stored in the device’s NMIB. All RRP devices in a network share link information via NCMT messages. Every device updates its network information and path table when it receives a NCMT message containing network information. Every device on the same network shares and gathers link information on the network to update its own network information and path table. Every device updates its network information and path table when it receives link status change information.

4.7 Network recovery

When link failure or device failure is detected in a RRP ring network, the topology changes and the link status information is automatically broadcast to every device on the network. After broadcasting topology change information, every device on a network starts to update its own path table and tries to find new paths to other devices on the network. This process is operated in protocol machine and changing the blocking point of the network. Thus, devices can transmit messages to their destinations, while they are updating their NMIB.

4.8 Automatic network configuration

RRP supports automatic network configuration. When the network topology changes, the RRP protocol machine of every device shares the changed network information, and then every device updates its own NMIB. RNMs or LNMs are automatically selected on the network according to the device UID and connection status.

When a device joins to a RRP network, the device broadcasts a NCMT message with its network information. Other devices in a RRP network receive the NCMT message then update their NMIB and reply to a new device. Thus NMIB is updated automatically through a joining process.

Similarly when a device detects any fault conditions, the device broadcasts a NCMT message to other devices in the network. So that network information is managed automatically by NCMT messages.

4.9 RRP basic operating principle

The RRP network is established by following steps:

- power On (Initialization);
- establishment of line network;
- extension of line network;
- establishment of ring network;
- change topology from ring network to line network.

In the initialization phase, a RRP device becomes a stand-alone device. The device tries to find valid RRP connections on its ports. Figure 6 shows that Device1 is initialized and remained stand-alone device. In this phase, the device is in SA state.

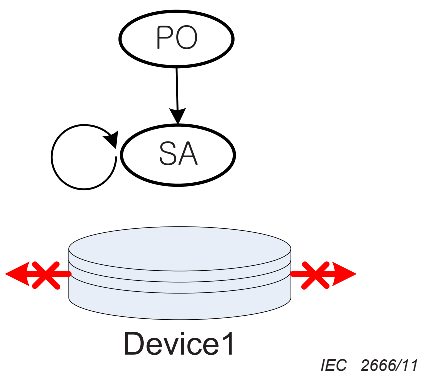

Figure 6 – A device operation in initialization phase

When the device detects a valid RRP connection on any port, it changes its state from SA to LNM and broadcast a NCMT message to the network. Figure 7 shows the device connection and its state.

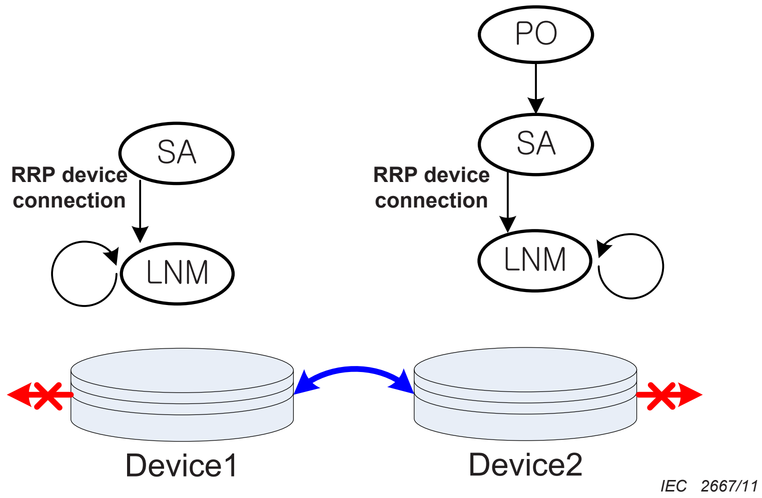

Figure 7 – Devices operation in line network establishing phase

If one or more devices added to the line network, the network is extended and two devices at both ends of the line network remain LNMs. This process is operated by RRP protocol machine in each device with NMIB. Figure 8 shows extension of line network operation.

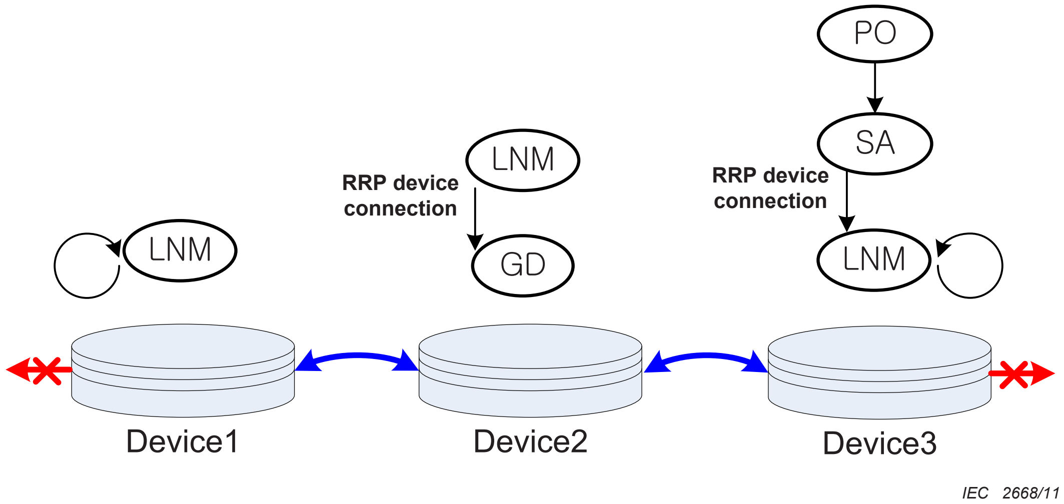

Figure 8 – Extension of line network operation

When both ends of the line network are connected, the line network is changing to ring network. Then the highest UID device sends a NCMT message as a RNMP. RNMP chooses R-port1 side neighbouring device as RNMS. Figure 9 shows a ring network establishment operation.

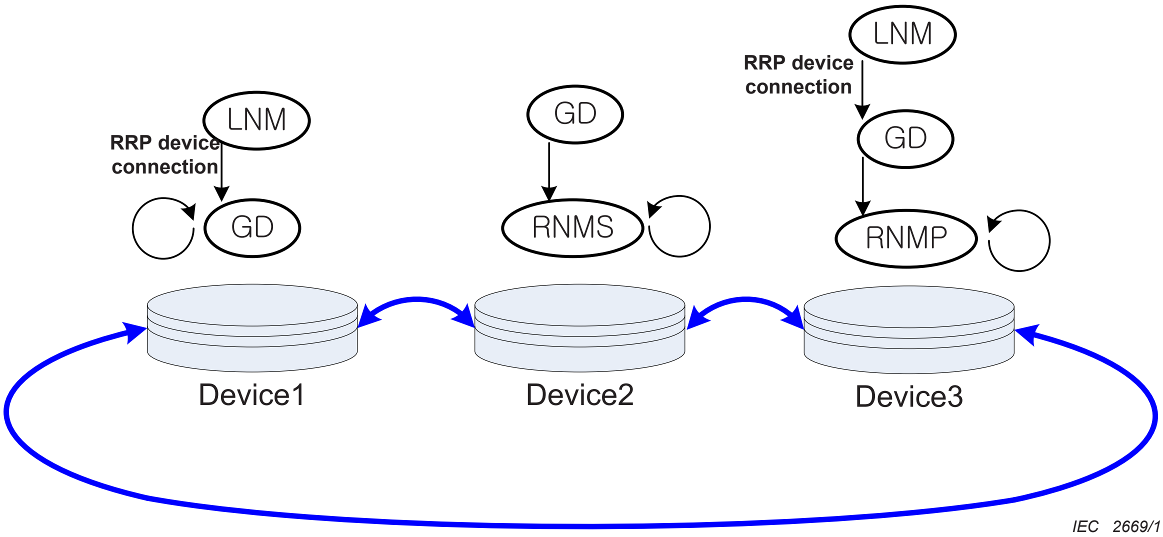

Figure 9 – Ring network establishment operation

If any fault condition is detected, neighbouring devices of the fault point broadcast NCMT message and change their state to LNM. Then the ring network is changed to line network. Figure 10 shows topology changing operation from ring network to line network. If the loss of valid connection is recovered, the network will be changed to ring network automatically as shown Figure 9.

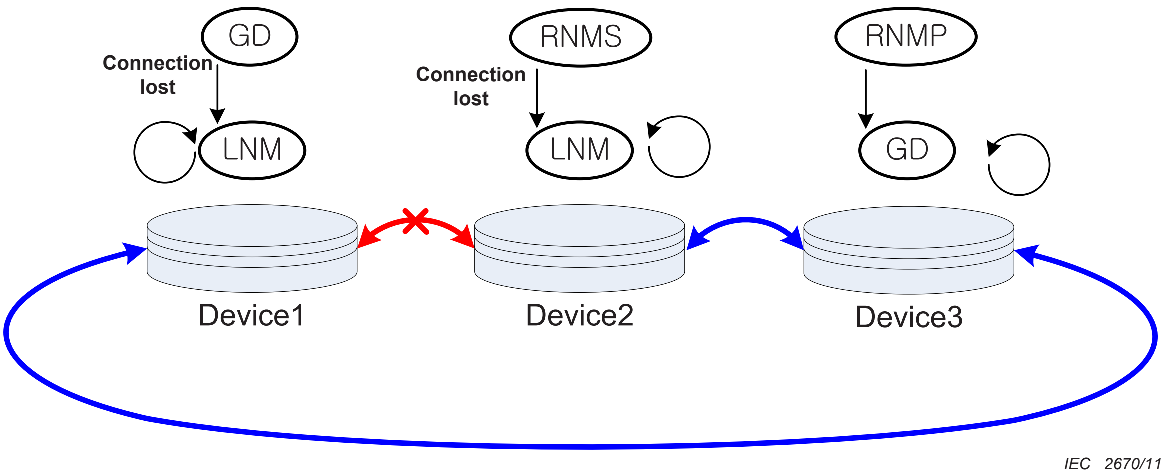

Figure 10 – Ring to line network change operation

5 RRP redundancy behaviours

5.1 Network topology

Figure 11 shows a basic example of a RRP ring network.

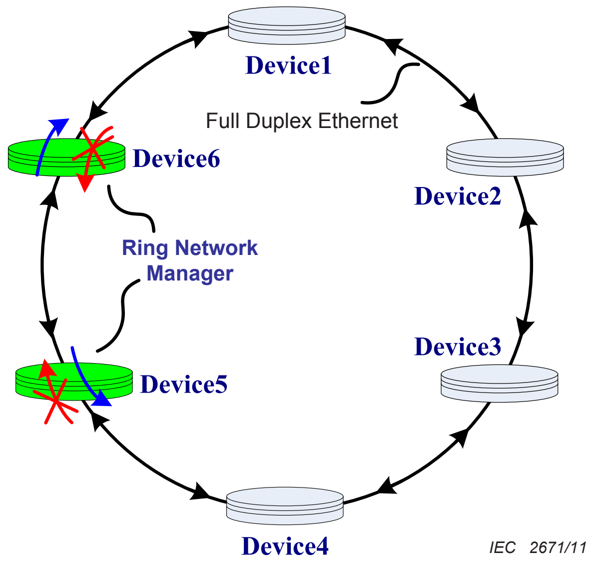

Figure 11 – Ring topology

When a new link is established between two end nodes in a line network, the network is automatically reconfigured as a ring network. Two RNMs are automatically selected to prevent the infinite circulation of any frame in a ring network. However, when a link failure is detected in the ring network, the ring network managers (RNMs) change their internal state to convert network topology from a ring to a line network. RNM could send NCMT message periodically to other RNM though the network to examine of the network integrity.

5.2 Network recovery in ring network

5.2.1 General

The RRP network recovery process is simple and is handled within the protocol machine to provide fast recovery time upon network topology changes. When a link failure is detected in a ring network, the network is reconfigured automatically as a line network. A topology change from a ring to a line network occurs in the following three cases:

- link fault with the neighbouring device;
- link fault of the remote device;
- local device fault.

If a link failure occurs, the two devices nearest the fault point enter the LNM state and broadcast the NCMT message NCM_LINE_START. When a device receives NCM_LINE_START, it checks its connection status and changes its state to GD or LNM.

A cable break or power failure is treated in the same way as a link fault. A device fault also takes the same time to recover. Redundancy recovery time is determined from the time when a link fault occurs to the time when the path is recovered in a new direction.

The RRP specifies recovery times for the transition from a ring to a line network. Table 1 shows the parameter set for worst case RRP recovery times when the network parameter is set as Table 2.

Table 1 – RRP network recovery parameter

Parameter	100BASE-X	1000BASE-X	Meaning
T_fault_sense	350 us	2 ms	the network fault sense time in PHY specified in ISO/IEC 8802-3:2000, Clause 24, Clause 25, Clause 26 and Clause 36 100BASE-TX: ANSI X3.263-1995, Clause 10 100BASE-FX: ISO/IEC 9314-3:1990, Clause 9 1000BASE-X: ANSI X3.230-1994, Annex I
T_state_transient	1 ms	1 ms	the time spent switching the protocol machine from RNM or GD to LNM, in the RRP device
T_message_propagation	6 ms	700 µs	the message delivery time of NCM_LINE_START from a new LNM to the farthest device in a segment
T_recovery (max.)	≤ 8 ms	≤ 4 ms	the total network recovery time in ms

Table 2 – Parameters for calculation

Parameter	Value		Meaning
	100BASE-X	1000BASE-X	
N	50	50	the number of nodes between sending and receiving devices
T_{PKT}	24 µs	2,4 µs	the packet transmit time
NCMsize	112 octets	112 octets	the message length of NCM_LINE_START message
POsize	40 octets	40 octets	the size of protocol overhead
LDR	100	1 000	the link speed in Mbit/s
T_{CPD}	0,5 µs	0,05 µs	the cable propagation time of node for 100 m
T_{SND}	50 µs	50 µs	the sender stack traversal time including PHY and MAC
T_{RCV}	50 µs	50 µs	the receiver stack traversal time including PHY and MAC
T_{NLD}	120 µs	12 µs	the node latency delay time for worst case
	3 µs	0,3 µs	the node latency delay time for best case

The network recovery time can be calculated by Equation (1)

$$T_{RECOVER} = T_{FS} + T_{RtoL} + T_{MSG} \tag{1}$$

where

$T_{RECOVER}$ is the network recovery time in ms;

T_{FS} is the link fault sense time in ms;

T_{RtoL} is the device state transition delay time in µs;

T_{MSG} is the message propagation time to the farthest device of NCM_LINE_START message in ms, see Equation (2).

The NCMT message, NCM_LINE_START, propagation time to the farthest device T_{MSG} can be calculated by Equation (2).

$$T_{MSG} = T_{SND} + T_{PKT} + T_{CPD} + \sum_{i=0}^{N} T_{NLD\_i} + T_{RCV} \tag{2}$$

where

T_{MSG}	is the message propagation time in µs;
T_{SND}	is the sender stack traversal time including PHY and MAC in µs;
T_{PKT}	is the packet transmit time in microseconds, see Equation (3);
T_{CPD}	is the cable propagation time for a node, in µs;
T_{NLD_i}	is the node latency time for node i in micro seconds, see Equation (4);
T_{RCV}	is the receiver stack traversal time including PHY and MAC in µs;
N	is the number of nodes between sending and receiving devices.

The packet transmit time T_{PKT} can be calculated by Equation (3)

$$T_{PKT} = \frac{(\text{NCMsize} + \text{POsize}) \times 8}{\text{LDR}} \tag{3}$$

where

T_{PKT}	is the packet transmit time on the medium, in µs;
NCMsize	is the size of the NCM data unit in octets;
LDR	is the link data rate in bits per second;
POsize	is the size of the protocol overhead in octets.

The node latency time of node i T_{NLD_i} can be calculated by Equation (4)

$$T_{NLD\_i} = T_{NPD\_i} + T_{PKT\_i} + \sum_{j=0}^{M} T_{TX\_PKT\_ij} \tag{4}$$

where

T_{NLD_i}	is the node latency delay time for node i in µs;
T_{NPD_i}	is the node propagation delay time for node i in µs;
T_{PKT_i}	is the packet transmit time for node i in µs;
$T_{TX\_PKT\_ij}$	is the packet transmit time for packet j in ms within the port transmit queue of node i in front of this packet;
M	is the number of packets in the port transmit queue of node i in front of this packet.

5.2.2 Link fault between neighbouring devices

Figure 12 shows an example of a link fault between neighbouring devices in a ring network.

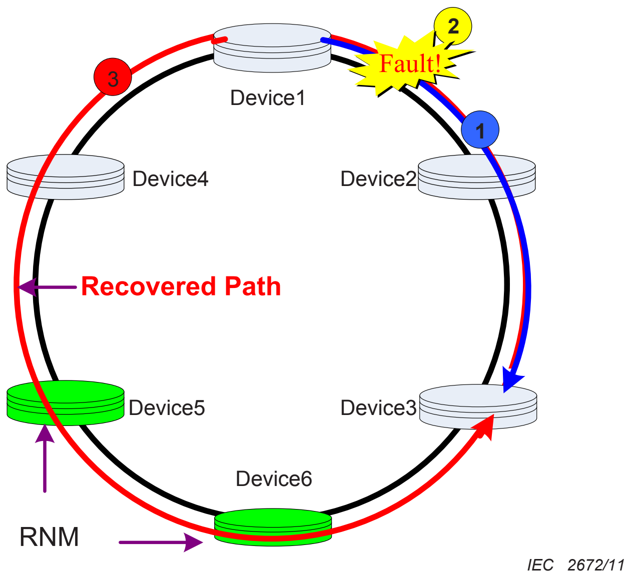

Figure 12 – Link fault between neighbouring devices

If the link between Device1 and Device2 is disconnected when Device1 tries to send a frame to Device3, the link fault event is triggered spontaneously by a hardware signal, and it is detected by the RRP protocol machine in Device1. Device1 then decides that the link is not available for data transmission, and so the ring network should be reconfigured as a line network, by changing the state of Device1 to LNM. Device1 broadcasts a NCM_LINE_START message and modifies the destination R-port to Device3. Device1 transmits the frame to the new destination R-port. Device2 also broadcasts a NCM_LINE_START message, to announce the link fault, and changes its own state to LNM.

In case that Device2 could not detect the link fault, Device5 receives NCM_LINE_START message from Device1 and change its state from RNM to GD. Then, Device5 relays NCM_LINE_START message to Device6. Thus, other devices of the network recognize topology changing.

5.2.3 Link fault of remote device

Figure 13 shows an example of a link fault of the remote device in a ring network.

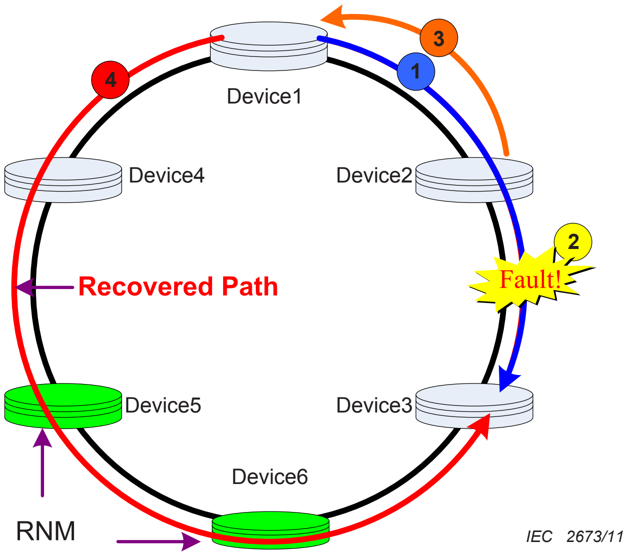

Figure 13 – Link fault of remote device

If the link between Device2 and Device3 is disconnected when Device1 tries to send a frame to Device3, the link fault event is triggered spontaneously by the hardware signal and is detected by the RRP protocol machine in Device2. At this point, Device2 broadcasts an NCM_LINE_START message indicating that the link is not available and that the ring network should be reconfigured as a line network. Device1 modifies the destination R-port to Device3 as the other R-port and transmits the frame through the new R-port. Device3 also broadcasts a NCM_LINE_START message, to announce the link fault, and changes its own state to LNM.

5.2.4 Device fault on a RNM

Figure 14 shows an example of a device fault on a RNM in a ring network.

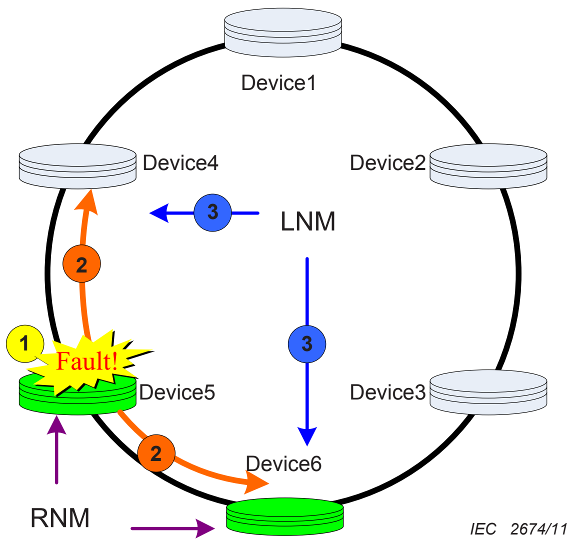

Figure 14 – Device fault on a RNM

If a fault occurs on Device5, Device6 and Device4 sense link failure on their respective Device5 side R-ports. Device6 and Device4 both broadcast NCM_LINE_START messages to other devices and change their state of to LNM. A GD fault is treated in same way as a RNM fault.

5.3 Automatic Ring Network Manager (RNM) election procedure

5.3.1 General

To prevent infinite frame circulation in a RRP ring network, the primary RNM (RNMP) and the secondary RNM (RNMS) are selected automatically using their device unique IDs (UID), via the following steps:

- when two LNMs are connected, each LNM sends NCMT message as multicast or broadcast;
- every device which receives NCMT message responds with its UID as multicast or broadcast;
- when a device receives its own responded NCMT message, the device detects the network is a ring network and does not forward the message;
- the device with the highest device UID value in the ring network is selected automatically as the RNMP;
- the RNMP sends an NCM_RING_START message including the RNMS assignment request to the neighbouring device connected through its both of R-port1 and R-port2;
- the RNMS replies with an NCM_ACK_RNMS message to the RNMP;
- automatic ring network configuration is completed with RNMP and RNMS;
- the RNMP blocks frame forwarding to RNMS side R-port and RNMS also blocks frame forwarding to RNMP side R-port to prevent looping on the ring.

5.3.2 Primary RNM (RNMP)

A RRP device realizes that the network is configured as a ring topology when the NCMT message generated by the device itself is received through the other port. In a ring network, the device with the highest device UID value is selected as the RNMP. When a RRP device detects that the network is configured as a ring network, each device tries to find the device in the path table that has the highest device UID value. Thus, competition to select the RNMP is not necessary in RRP. If a remote device has the highest UID on the network, the other devices wait for the NCM_RING_START message from it. If a local device has the highest UID on the network, then it is selected as the RNMP, the RRP protocol machine disables both frame forwarding functions to prevent looping on the network, and generates an NCM_RING_START message, including the RNMS information of the device connected through both of R-port1 and R-port2 of the RNMP.

When the RNMP receives the NCM_ACK_RNMS message from the RNMS, it disables the frame forwarding function in the RNMS direction, but keeps the opposite direction enabled. The RNMS disables the frame forwarding function in the RNMP direction and enables it in the opposite direction.

RNMP could send NCM_CHECK_RNMS message to RNMS as a network integrity check frame periodically. If the response of NCM_CHECK_RNMS message is not arrived within the timer RRP_ChkRNMST from RNMS and exceed the number of network integrity retry count, the RNMP figures out the network has an error.

NOTE 1 The device UID has a unique value on the network. The RNMP and RNMS are therefore selected automatically even in the case of a device address collision (See 5.5).

NOTE 2 NCM_CHECK_RNMS and NCM_ACK_RNMS messages are using to choose RNMS and to confirm RNMS by RNMP. But RRP supports these messages could be used to check network integrity by user. The timer and the retry count are specified as local variables (See Table 32).

5.3.3 Secondary RNM (RNMS)

The RNMS is assigned by the RNMP. When a GD device receives an NCM_RING_START message from the RNMP, the RRP protocol machine compares the local device UID to the RNMS device UID in the received NCM_RING_START message. If the device is not designated as the RNMS, the RRP protocol machine enables both frame forwarding functions in the GD state device. If the device is designated as the RNMS, the RRP protocol machine enters the RNMS state and transmits NCM_ACK_RNMS to the RNMP through the received R-port of the NCM_RING_START message and NCM_CHECK_RNMS message from the RNMP. Additionally, the RNMS device disables the frame forwarding function in the RNMP direction, but enables it in the opposite direction.

5.4 Path management

5.4.1 General

The RRP device provides path information about each RRP device on the network. Path management is calculated from the hop count. The hop count indicates how many frame forward operations are required to transfer a frame to the destination device.

5.4.2 Path in a line topology network

In a line network, only one path is possible to the destination device. Figure 15 shows an example of path management in a line topology network.

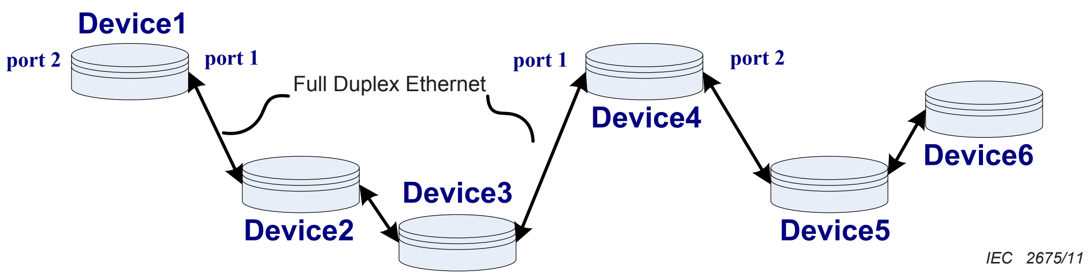

Figure 15 – Path management in a line topology network

Table 3 shows the path table of Device1 and Table 4 shows the path table of Device4 in Figure 15. In a line network, only one path is possible between any two devices.

Table 3 – Path table of Device1 in a line topology network

R-port	Destination				
	Device2	Device3	Device4	Device5	Device6
R-port1	0 hop	1 hop	2 hops	3 hops	4 hops
R-port2	Invalid	Invalid	Invalid	Invalid	Invalid
Preferred Port	R-port1	R-port1	R-port1	R-port1	R-port1
Destination Port	R-port1	R-port1	R-port1	R-port1	R-port1

Table 4 – Path table of Device4 in a line topology network

R-port	Destination				
	Device1	Device2	Device3	Device5	Device6
R-port1	2 hops	1 hop	0 hop	Invalid	Invalid
R-port2	Invalid	Invalid	Invalid	0 hop	1 hop
Preferred Port	R-port1	R-port1	R-port1	R-port2	R-port2
Destination Port	R-port1	R-port1	R-port1	R-port2	R-port2

5.4.3 Path in a ring topology network

In a ring network, two paths are possible between any two devices: the clockwise path and the counter clockwise path. However, a frame cannot be forwarded across the RNMP or RNMS. Therefore, it is impossible to transfer a frame using the path including the RNMP or RNMS. Figure 16 shows an example of path management in a ring network.

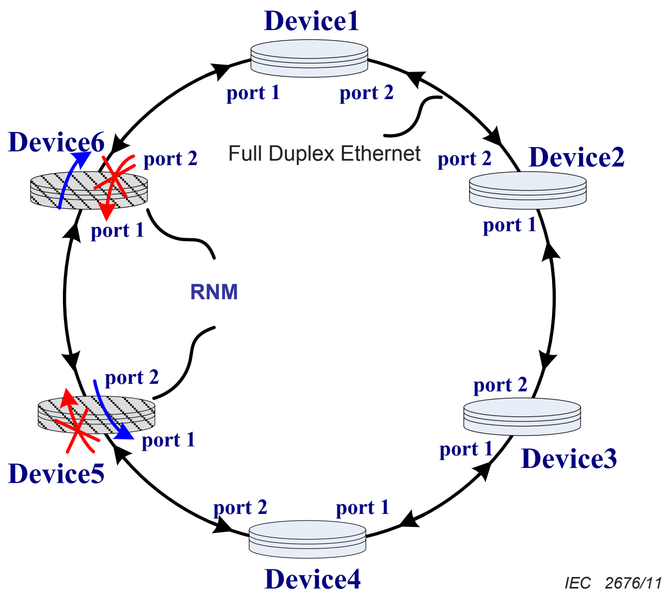

Figure 16 – Path management in a ring topology network

Table 5 shows the path table of Device1 in Figure 16. The shortest path from Device1 to Device5 is in the R-port1 direction, but this path is blocked by the RNM, Device6. In this case, the destination path is determined to be in the R-port2 direction.

Table 5 – Path table of Device1 in a ring topology network

R-port	Destination				
	Device2	Device3	Device4	Device5 (RNM)	Device6 (RNM)
R-port1	4 hops	3 hops	2 hops	1 hop	0 hop
R-port2	0 hop	1 hop	2 hops	3 hops	4 hops
Preferred Port	R-port2	R-port2	Don't care	R-port1 (R-port1 direction is blocked by Device6)	R-port1
Destination Port	R-port2	R-port2	R-port1(NOTE)	R-port2	R-port1
NOTE If both paths have the same hop counts and they are not blocked by the RNMs, the R-port1 direction is always chosen.					

Table 6 shows the path table of Device3 in Figure 16.

Table 6 – Path table of Device3 in a ring topology network

R-port	Destination				
	Device1	Device2	Device4	Device5 (RNM)	Device6 (RNM)
R-port1	3 hops	4 hops	0 hop	1 hop	2 hops
R-port2	1 hop	0 hop	4 hops	3 hops	2 hops
Preferred Port	R-port2	R-port2	R-port1	R-port1	Don't care (R-port1 direction is blocked by Device5)
Destination Port	R-port2	R-port2	R-port1	R-port1	R-port2

5.5 Device address collision

A RRP device address is configured manually by hardware settings (e.g., rotary switch) or set by software (e.g., portable terminal). The device address may be duplicated in other devices on the network because of a configuration error. When this happens, the device is unable to communicate with other devices. This type of device address collision event on the network is detected automatically by the RRP protocol machine.

RRP uses 8 octets UID instead of device address to manage network resources. UID is made combination of device address and MAC address so that UID is a unique value in the network. Network and UID information is stored in NMIB of each device. Thus, devices of network could know which device has duplicated address.

Figure 17 shows an example of device address collision in a ring network.

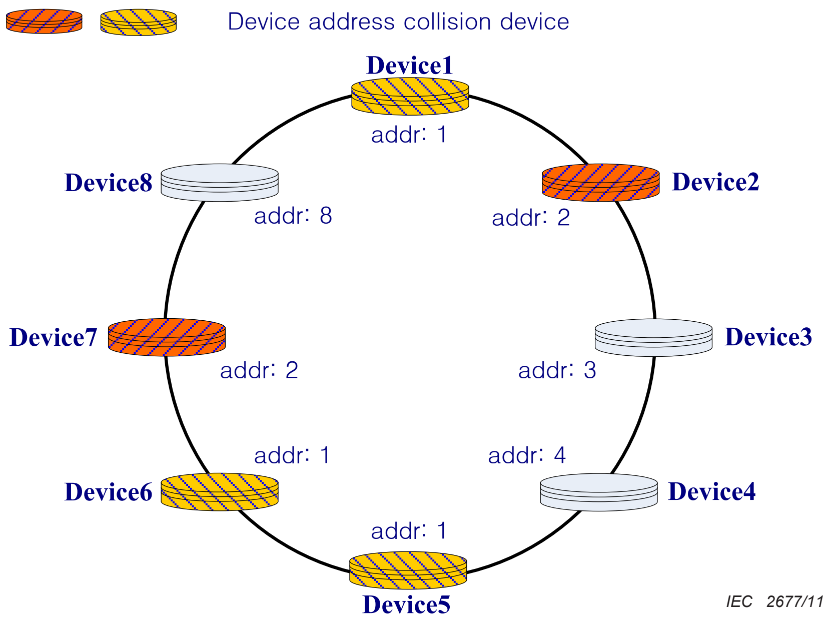

Figure 17 – RRP device address collision in a ring network

Devices1, 5, and 6 have the same device address value of 1. Device2 and Device7 have the same device address value of 2. Therefore, three device address collision events are detected by the RRP protocol machine. Device address collision events are counted by the collision counters in the network management information base and shared with every device on the network.

Table 7 shows the device address collision information for the situations in Figure 17 representing the device address collision detection mechanism using the device UIDs.

Table 7 – Device address collision information

Device name	Device address	MAC address	Device UID	Collision
Device1	0x0001	0x002233445511	0x0001002233445511	Collision
Device2	0x0002	0x002233445522	0x0002002233445522	Collision
Device3	0x0003	0x002233445533	0x0003002233445533	—
Device4	0x0004	0x002233445544	0x0004002233445544	—
Device5	0x0001	0x002233445555	0x0001002233445555	Collision
Device6	0x0001	0x002233445566	0x0001002233445566	Collision
Device7	0x0002	0x002233445577	0x0002002233445577	Collision
Device8	0x0008	0x002233445588	0x0008002233445588	—

The device address is configured manually by the operator to be some value in the range 0-255. The MAC address is a 6 octet unique ISO/IEC 8802-3:2000 Ethernet MAC address. The RRP device UID contains 2 octet RRP device address and the 6 octet MAC address. Therefore, the RRP device UID has a unique 8 octet value on the network that cannot be duplicated. The RRP device UID is used to recognize a specific device on the network.

A RRP device address collision is detected by the RRP protocol machine when devices with different device UIDs are configured with the same device address. If a local device address collision is detected, the device sets the device address collision flag in the local device flags and generates a device address collision event. If a remote device address collision is detected, the device sets the device address collision flag in the network flags and generates a network device address collision event.

6 RRP class specification

6.1 General

The RRP Application Service Element (ASE) defines one object type.

6.2 Template

A RRP object is described by the following template:

ASE: **Ring-based redundancy ASE**
CLASS: **Ring-based redundancy**
CLASS ID: **not used**
PARENT CLASS: **IEEE 802.3 Ring-based Redundancy Protocol**
ATTRIBUTES:

1	(m)	Key Attribute:	Local device address
2	(m)	Attribute:	Local device flags
3	(m)	Attribute:	Local device state
4	(m)	Attribute:	Local device unique ID
5	(m)	Attribute:	Device UID for R-port1
6	(m)	Attribute:	Device UID for R-port2
7	(m)	Attribute:	Local device MAC address
8	(m)	Attribute:	Local device R-port1 information

9	(m)	Attribute:	Local device R-port2 information
10	(m)	Attribute:	Local device protocol version
11	(m)	Attribute:	Local device type
12	(m)	Attribute:	Local device description
13	(m)	Attribute:	Hop count
14	(m)	Attribute:	FamilyRes frame waiting time
15	(m)	Attribute:	AdvThis frame waiting time
16	(m)	Attribute:	AckRNMS frame waiting time
17	(m)	Attribute:	Ring state change timeout
18	(m)	Attribute:	Diagnostic information (Network, Path Table)
19	(c)	Constraint:	Diagnostic information = Network
19.1	(m)	Attribute:	Network topology
19.2	(m)	Attribute:	Collision count
19.3	(m)	Attribute:	Device count
19.4	(m)	Attribute:	Topology change count
19.5	(m)	Attribute:	Last topology change time
19.6	(m)	Attribute:	RNMP device UID
19.7	(m)	Attribute:	RNMS device UID
19.8	(m)	Attribute:	LNM device UID for R-port1
19.9	(m)	Attribute:	LNM device UID for R-port2
19.10	(m)	Attribute:	Network flags
20	(c)	Constraint:	Diagnostic information = Path Table
20.1	(m)	Attribute:	Peer device address
20.2	(m)	Attribute:	Peer device hop count for R-port1
20.3	(m)	Attribute:	Peer device hop count for R-port2
20.4	(m)	Attribute:	Preferred R-port for peer device
20.5	(m)	Attribute:	Destination R-port for peer device
20.6	(m)	Attribute:	Peer device state
20.7	(m)	Attribute:	Peer device MAC address
20.8	(m)	Attribute:	Peer device R-port1 information
20.9	(m)	Attribute:	Peer device R-port2 information
20.1	(m)	Attribute:	Peer device protocol version
20.11	(m)	Attribute:	Peer device type
20.12	(m)	Attribute:	Peer device description
20.13	(m)	Attribute:	Peer device UID
20.14	(m)	Attribute:	Device UID for R-port1 of peer device
20.15	(m)	Attribute:	Device UID for R-port2 of peer device
20.16	(m)	Attribute:	In net count of peer device
20.17	(m)	Attribute:	In net time of peer device
20.18	(m)	Attribute:	Out net count of peer device
20.19	(m)	Attribute:	Out net time of peer device

SERVICES:

1	(m)	OpsService:	Set Device Information
2	(m)	OpsService:	Get Device Information
3	(m)	OpsService:	Get Network Information
4	(m)	OpsService:	Get Path Table Information

6.3 Attributes

NOTE 1 The data type of each attribute is following definition of the standard IEC 61158-4-21:2010, Clause 4.

Local device address
This key attribute defines the RRP local device address that designates a single RRP device on a specific local link; its value is constrained to the range 0-255.

Data type: Unsigned16

NOTE 2 The local device address may be provided by hardware settings (e.g., rotary switch) or set by software.

Local device flags
This attribute specifies the flags for events that occurred in a local device. The meaning for each bit is as follows:

- Bit 0: device address collision
- Bit 1: device state changed

Data type: Unsigned16

Local device state
This attribute specifies the local device state. The device state shall have one of the following values:

- 0 = invalid
- 1 = SA (standalone state)
- 2 = LNM (line network manager state)
- 3 = GD (general device state)
- 4 = RNMP (primary ring network manager state)
- 5 = RNMS (secondary ring network manager state)

Data type: Unsigned8

Local device MAC address
This attribute defines the 6 octet ISO/IEC 8802-3:2000 Ethernet MAC address of the local device. Because a RRP device has two Ethernet MAC ports, both MAC addresses should be identical.

Data type: Unsigned48

Local device unique ID
This attribute defines the unique 8 octet identification that identifies a RRP device in a network. It is a combination of the 6 octet ISO/IEC 8802-3:2000 MAC address and the 2 octet device address. The individual bits shall have the following meaning:

- Bit 0 – 15: device address
- Bit 16 – 63: ISO/IEC 8802-3:2000 MAC Address

Data type: UniqueDeviceID64

Device UID for R-port1
This attribute defines the UID of the device that is linked through the R-port1. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

Device UID for R-port2
This attribute defines the UID of the device that is linked through the R-port2. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

Local device R-port1 information
This attribute defines the port information for R-port1. The meaning for each bit is as follows:

- Bit 0: R-port link down
- Bit 1: received Family_Frame.Cnf from R-port
- Bit 2: waiting for AdvThis_Frame.Cnf from R-port
- Bit 3: waiting for MediaLinked_Frame.Ind from R-port
- Bit 4: device state confirm

Data type: Unsigned8

Local device R-port2 information
This attribute defines the port information for R-port2. It shall have one of the values defined for local device R-port1 information.

Data type: Unsigned8

Local device protocol version
This attribute defines the RRP protocol version of the local device. The individual bits shall have the following meaning:

- Bit 0 – 1: major version
- Bit 2 – 4: minor version
- Bit 5 – 7: reserved

Data type: Unsigned8

Local device type
This attribute defines the local device type that represents the general function of the device. The individual bits shall have the following meaning:

- Bit 0 – 7: general device type
- Bit 8 – 15: application-specific device type

Data type: Unsigned16

Local device description
This attribute defines a description of the local device and contains any string defined by the user using the Set Device Information service.

Data type: VisibleString[16]

Hop count
This attribute defines the count of the number of devices between two devices. When the RRP device receives the NCM frame, the RRP device saves the received hop count value in this variable, then increments the hop counts in the received frame by 1 and transmits the frame through the other R-port. In this way, each device builds its own path table with a hop count for R-port1 and a hop count for R-port2.

Data type: Unsigned16

FamilyRes frame waiting time
This attribute defines the time interval between sending the FamilyReq frame and receiving the FamilyRes frame. This attribute shall be configured by the user using the Set Device Information service.

Data type: Unsigned32

AdvThis frame waiting time
This attribute defines the time interval between sending the MediaLinked frame and receiving the AdvThis frame. This attribute shall be configured by the user using the Set Device Information service.

Data type: Unsigned32

AckRNMS frame waiting time
This attribute defines the time interval between sending the RingStart frame and receiving the AckRNMS frame. This attribute shall be configured by the user using the Set Device Information service.

Data type: Unsigned32

Ring state change timeout
This attribute defines the timeout to generate the event for changing the RNMP device state. This attribute shall be configured by the user using the Set Device Information service.

Data type: Unsigned32

Diagnostic information
This attribute defines the type of diagnostic information. The type of diagnostic information shall have the two values of Network and Path Table.

Data type: Unsigned8

Network topology
This attribute defines the type of network topology. It shall have one of the following values:

- 0 = invalid
- 1 = NET_TPG_SA (standalone)
- 2 = NET_TPG_LINE (line topology)
- 3 = NET_TPG_RING (ring topology)

Data type: Unsigned8

Collision count
This attribute defines the device address collision count for remote devices. Values are in the range 0-255.

Data type: Unsigned8

NOTE 3 The value is incremented by the RRP protocol machine when a remote device address collision is detected, and the value is decremented when the collision is cleared.

Device count
This attribute defines the total number of devices on the network. Values are in the range 1-255.

Data type: Unsigned16

Topology change count
This attribute defines the topology change count. The values are in the range 0-65 535.

Data type: Unsigned16

NOTE 4 The value is incremented by the RRP protocol machine when the network changes from ring to line or from line to ring topology.

Last topology change time
This attribute defines the date and time when the network topology was last changed.

Data type: TIMEOFDAY

RNMP device UID
This attribute defines the device UID selected as the RNMP on the network. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

RNMS device UID
This attribute defines the UID of the device selected as the RNMS on the network. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

LNM device UID for R-port1
This attribute defines the UID of the device selected as the LNM in the R-port1 direction. In a RRP line network, the two end devices are automatically selected as the LNMs. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

LNM device UID for R-port2
This attribute defines the UID of the device selected as the LNM in the R-port2 direction. In a RRP line network, the two end devices are automatically selected as the LNMs. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

Network flags
This attribute defines the flags for events that occurred in the network. The meaning for each bit is as follows:

- Bit 0: network topology has changed
- Bit 1: device address collision has been detected in the network
- Bit 2: new device has joined the network
- Bit 3: device has left the network

Data type: Unsigned16

NOTE 5 Each bit is set by the RRP protocol machine when the corresponding event occurs.

Peer device address
This attribute defines the peer device address stored in the path table. The value is constrained to the range 0-255.

Data type: Unsigned16

Peer device hop count for R-port1
This attribute defines the frame forwarding counts for sending a frame from the local device to the peer device through the R-port1.

Data type: Unsigned16

Peer device hop count for R-port2
This attribute defines the frame forwarding counts for sending a frame from the local device to the peer device through the R-port2.

Data type: Unsigned16

Preferred R-port for peer device
This attribute defines the preferred R-port for sending a frame from the local device to the peer device without regard for the RNMP or RNMS. It shall have one of the following values:

- 0 = invalid
- 1 = R-port1
- 2 = R-port2

Data type: Unsigned8

NOTE 6 It is defined as the R-port that has the smaller hop count value for the peer device. If the R-port1 and R-port2 hop counts have the same value, R-port1 is selected as the preferred R-port.

Destination R-port for peer device
This attribute defines the destination R-port for sending a frame from the local device to the peer device.

Data type: Unsigned8

NOTE 7 In a line network, this variable has the same value as the preferred R-port. However, in a ring network, this variable is determined based on the RNMP and RNMS positions, because the preferred path may be blocked by the RNMP or RNMS. In this case, the destination R-port is selected as the other R-port. It should have one of the values defined for the preferred R-port.

Peer device state
This attribute defines the peer device state stored in the path table. It shall have one of the values defined for the local device state.

Data type: Unsigned8

Peer device MAC address
This attribute defines the peer device 6 octets ISO/IEC 8802-3:2000 Ethernet MAC address stored in the path table.

Data type: Unsigned48

Peer device R-port1 information
This attribute defines the peer device R-port1 information stored in the path table. It shall have one of the values defined for the local device R-port1 information.

Data type: Unsigned8

Peer device R-port2 information
This attribute defines the peer device R-port2 information stored in the path table. It shall have one of the values defined for the local device R-port1 information.

Data type: Unsigned8

Peer device protocol version
This attribute defines the peer device RRP protocol version stored in the path table. The individual bits shall have the same meaning as the local device protocol version.

Data type: Unsigned8

Peer device type
This attribute defines the peer device application device type stored in the path table. The individual bits shall have the same meaning as the local device type.

Data type: Unsigned16

Peer device description
This attribute defines the peer device description stored in the path table.

Data type: VisibleString[16]

Peer device UID
This attribute defines the peer device unique ID stored in the path table. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

Device UID for R-port1 of peer device
This attribute defines the UID of the device that is linked through the R-port1 of the peer device and stored in the path table. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

Device UID for R-port2 of peer device
This attribute defines the UID of device that is linked through the R-port2 of the peer device and stored in the path table. The individual bits shall have the same meaning as the local device unique ID.

Data type: UniqueDeviceID64

In net count of peer device
This attribute defines the number of times that the peer device has joined the network

Data type: Unsigned16

NOTE 8 When a line network is merged into an existing network, the variables for the newly joined devices are incremented together.

In net time of peer device
This attribute defines the date and time when the peer device last joined in the network.

Data type: TIMEOFDAY

Out net count of peer device
This attribute defines the number of times that the peer device has been disconnected from the network. When a device or a group of devices is disconnected from the network, the variables for the disconnected devices are incremented together.

Data type: Unsigned16

Out net time of peer device
This attribute defines the date and time when the device was last disconnected from the network.

Data type: TIMEOFDAY

7 RRP services specification

7.1 Set device information

This service is used to assign new values to the variables of the local device information.

The parameters of this service are specified in Table 8.

Table 8 – Parameters of set device information service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Service ID	M	M(=)		
Invoke ID	M	M(=)		
Local device address	M	M(=)		
Local device MAC address	M	M(=)		
Local device protocol version	M	M(=)		
Local device type	M	M(=)		
Local device description	M	M(=)		
FamilyRes frame waiting time	M	M(=)		
AdvThis frame waiting time	M	M(=)		
AckRNMS frame waiting time	M	M(=)		
Ring state change timeout	M	M(=)		

Parameter name	Req	Ind	Rsp	Cnf
Result(+)			S	S(=)
Service ID			M	M(=)
Invoke ID			M	M(=)
Status Code			M	M(=)
Result(-)			S	S(=)
Service ID			M	M(=)
Invoke ID			M	M(=)
Status Code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

Service ID
This parameter contains information sufficient for local identification of the RRP device to be used to convey the service.

Invoke ID
This parameter identifies this invocation of the service.

Local device address
This parameter contains the value for the RRP device address.

Local device MAC address
This parameter contains the MAC address of the RRP device.

Local device protocol version
This parameter contains the protocol version of the RRP device.

Local device type
This parameter contains the RRP device type that represents the general function of the RRP device.

Local device description
This parameter contains a description of the RRP device.

FamilyRes frame waiting time
This parameter contains the value for the FamilyRes frame waiting time of the RRP device.

AdvThis frame waiting time
This parameter contains the value for the AdvThis frame waiting time of the RRP device.

AckRNMS frame waiting time
This parameter contains the value for the AckRNMS frame waiting time of the RRP device.

Ring state change timeout
This parameter contains the value for the ring state change timeout of the RRP device.

Result(+)
This parameter indicates that the service request succeeded.

Service ID
This parameter contains information sufficient for local identification of the RRP device to be used to convey the service.

Invoke ID
This parameter identifies this invocation of the service.

Status Code
This parameter indicates whether the service was processed successfully. If an error occurred, it indicates the type of error.

Data type: Unsigned8

Result(-)
This parameter indicates that the service request failed.

Service ID
This parameter contains information sufficient for local identification of the RRP device to be used to convey the service.

Invoke ID
This parameter identifies this invocation of the service.

Status Code
This parameter indicates whether the service was processed successfully. If an error occurred, it indicates the type of error.

Data type: Unsigned8

7.2 Get device information

This service is used to obtain the local device information from the RRP device.

The parameters of this service are specified in Table 9.

Table 9 – Parameters of get device information service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Service ID	M	M(=)		
Invoke ID	M	M(=)		
Result(+)			S	S(=)
Service ID			M	M(=)
Invoke ID			M	M(=)
Local device address			M	M(=)
Local device flags			M	M(=)
Local device state			M	M(=)
Local device unique ID			M	M(=)
Device UID for R-port1			M	M(=)
Device UID for R-port2			M	M(=)
Local device MAC address			M	M(=)
Local device R-port1 information			M	M(=)
Local device R-port2 information			M	M(=)
Local device protocol version			M	M(=)
Local device type			M	M(=)
Local device description			M	M(=)
FamilyRes frame waiting time			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
AdvThis frame waiting time			M	M(=)
AckRNMS frame waiting time			M	M(=)
Ring state change timeout			M	M(=)
Result(-)			S	S(=)
Service ID			M	M(=)
Invoke ID			M	M(=)
Status Code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Result(+)
This parameter indicates that the service request succeeded.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Local device address
This parameter is defined in 7.1.

Local device flags
This parameter contains the device flags of the RRP device.

Local device state
This parameter contains the device state of the RRP device.

Local device unique ID
This parameter contains the device unique ID of the RRP device.

Device UID for R-port1
This parameter contains the UID of the device that is linked through the R-port1.

Device UID for R-port2
This parameter contains the UID of the device that is linked through the R-port2.

Local device MAC address
This parameter is defined in 7.1.

Local device R-port1 information
This parameter contains the R-port1 information of the RRP device.

Local device R-port2 information
This parameter contains the R-port2 information of the RRP device.

Local device protocol version
This parameter contains the protocol version of the RRP device.

Local device type
This parameter is defined in 7.1.

Local device description
This parameter is defined in 7.1.

FamilyRes frame waiting time
This parameter is defined in 7.1.

AdvThis frame waiting time
This parameter is defined in 7.1.

AckRNMS frame waiting time
This parameter is defined in 7.1.

Ring state change timeout
This parameter is defined in 7.1.

Result(-)
This parameter indicates that the service request failed.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Status Code
This parameter is defined in 7.1.

7.3 Get network information

This service is used to obtain network information from the RRP device.

The parameters of this service are specified in Table 10.

Table 10 – Parameters of get network information service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Service ID	M	M(=)		
Invoke ID	M	M(=)		
Result(+)			S	S(=)
Service ID			M	M(=)
Invoke ID			M	M(=)
Network topology			M	M(=)
Collision count			M	M(=)
Device count			M	M(=)
Topology change count			M	M(=)
Last topology change time			M	M(=)
RNMP device UID			M	M(=)
RNMS device UID			M	M(=)
LNM device UID for R-port1			M	M(=)
LNM device UID for R-port2			M	M(=)
Network flags			M	M(=)
Result(-)			S	S(=)

Parameter name	Req	Ind	Rsp	Cnf
Service ID			M	M(=)
Invoke ID			M	M(=)
Status Code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Result(+)
This parameter indicates that the service request succeeded.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Network topology
This parameter contains the type of network topology.

Collision count
This parameter contains the device address collision count for remote devices.

Device count
This parameter contains the total number of devices on the network.

Topology change count
This parameter contains the topology change count.

Last topology change time
This parameter contains the date and time when the network topology was last changed.

RNMP device UID
This parameter contains the UID of the device selected as the RNMP on the network.

RNMS device UID
This parameter contains the UID of the device selected as the RNMS on the network.

LNM device UID for R-port1
This parameter contains the UID of the device selected as the LNM in the R-port1 direction.

LNM device UID for R-port2
This parameter contains the UID of the device selected as the LNM in the R-port2 direction.

Network flags
This parameter contains the flags for events that occurred in the network.

Result(-)
This parameter indicates that the service request failed.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Status Code
This parameter is defined in 7.1.

7.4 Get path table information

This service is used to obtain path table information from the RRP device. The path table is managed by the RRP protocol machine in the form of an array table filled with the path information of each RRP device on the network. The maximum size of the path table is a function of MAX_ADDR as follows:

Path table: Array[n] of device's path information, n = MAX_ADDR + 1

NOTE The MAX_ADDR holds the maximum device address and is set by the RRP protocol machine. The range of this variable is 1-255. The default value of this variable is 255. This variable also indicates the maximum number of path table entries in the NMIB. The values in the range 256-65 535 are reserved.

The parameters of this service are specified in Table 11.

Table 11 – Parameters of get path table information service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Service ID	M	M(=)		
Invoke ID	M	M(=)		
Result(+)			S	S(=)
Service ID			M	M(=)
Invoke ID			M	M(=)
Peer device address			M	M(=)
Peer device hop count for R-port1			M	M(=)
Peer device hop count for R-port2			M	M(=)
Preferred R-port for peer device			M	M(=)
Destination R-port for peer device			M	M(=)
Peer device state			M	M(=)
Peer device MAC address			M	M(=)
Peer device R-port1 information			M	M(=)
Peer device R-port2 information			M	M(=)
Peer device protocol version			M	M(=)
Peer device type			M	M(=)
Peer device description			M	M(=)
Peer device UID			M	M(=)
Device UID for R-port1 of peer device			M	M(=)
Device UID for R-port2 of peer device			M	M(=)
In net count of peer device			M	M(=)
In net time of peer device			M	M(=)
Out net count of peer device			M	M(=)
Out net time of peer device			M	M(=)
Result(-)			S	S(=)

Parameter name	Req	Ind	Rsp	Cnf
Service ID			M	M(=)
Invoke ID			M	M(=)
Status Code			M	M(=)

Argument
The argument conveys the service specific parameters of the service request.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Result(+)
This parameter indicates that the service request succeeded.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Peer device address
This parameter contains the value for the RRP peer device address.

Peer device hop count for R-port1
This parameter contains the frame forwarding counts for sending a frame from the local device to the peer device through the R-port1.

Peer device hop count for R-port2
This parameter contains the frame forwarding counts for sending a frame from the local device to the peer device through the R-port2.

Preferred R-port for peer device
This parameter contains the preferred R-port for sending a frame from the local device to the peer device without regard for the RNMP or RNMS.

Destination R-port for peer device
This parameter contains the destination R-port for sending a frame from the local device to the peer device.

Peer device state
This parameter contains the peer device state stored in the path table.

Peer device MAC address
This parameter contains the peer device 6 octets ISO/IEC 8802-3:2000 Ethernet MAC address stored in the path table.

Peer device R-port1 information
This parameter contains the peer device R-port1 information stored in the path table.

Peer device R-port2 information
This parameter contains the peer device R-port2 information stored in the path table.

Peer device protocol version
This parameter contains the peer device RRP protocol version stored in the path table.

Peer device type
This parameter contains the peer device application device type stored in the path table.

Peer device description
This parameter contains the peer device description stored in the path table.

Peer device UID
This parameter contains the peer device unique ID stored in the path table.

Device UID for R-port1 of peer device
This parameter contains the UID of the device that is linked through the R-port1 of the peer device and stored in the path table.

Device UID for R-port2 of peer device
This parameter contains the UID of the device that is linked through the R-port2 of the peer device and stored in the path table.

In net count of peer device
This parameter contains the number of times that the peer device has joined the network.

In net time of peer device
This parameter contains the date and time when the peer device last joined the network.

Out net count of peer device
This parameter contains the number of times that the peer device has been disconnected from the network.

Out net time of peer device
This parameter contains the date and time when the device was last disconnected from the network.

Result(-)
This parameter indicates that the service request failed.

Service ID
This parameter is defined in 7.1.

Invoke ID
This parameter is defined in 7.1.

Status Code
This parameter is defined in 7.1.

8 RRP protocol specification

8.1 General

The encoding and decoding of the fields in the data link protocol data unit (DLPDU) is encapsulated in the data field of a MAC frame, as specified by ISO/IEC 8802-3:2000, Clause 3. The value of the Length/Type field is 0x88FE, which is authorized and registered as the protocol identification number by the IEEE Registration Authority to identify a RRP frame. Figure 18 shows the RRP DLPDU structure.

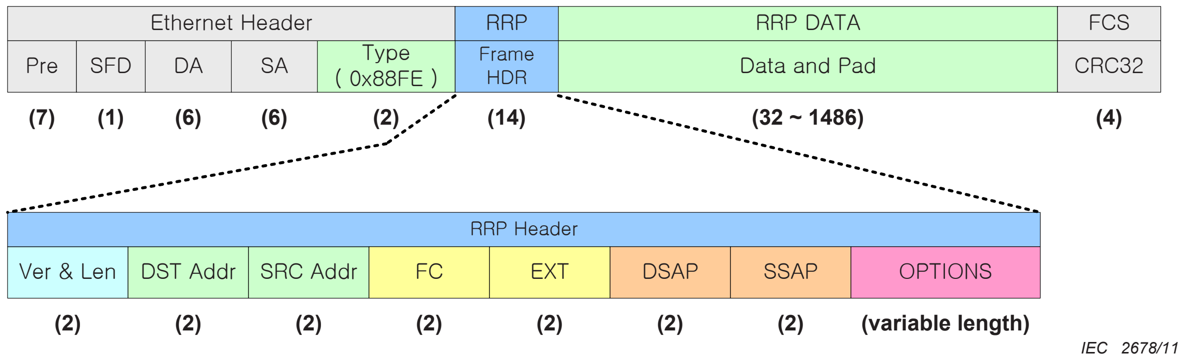

Figure 18 – Common MAC frame format for RRP DLPDU

8.2 Ethernet header

8.2.1 Preamble

This field shall be coded according to ISO/IEC 8802-3:2000, Clause 3.

8.2.2 Start frame delimiter

This field shall be coded according to ISO/IEC 8802-3:2000, Clause 3.

8.2.3 Destination MAC address

This field shall be coded according to ISO/IEC 8802-3:2000, Clause 3. It specifies the device(s) for which the frame is intended, and may be an individual or multicast (including broadcast) address.

8.2.4 Source MAC address

This field shall be coded according to ISO/IEC 8802-3:2000, Clause 3.

8.2.5 Length/Type

This field shall be coded according to ISO/IEC 8802-3:2000, Clause 3 “Media access control frame structure.” To be identified as a RRP frame, the value of the Length/Type field is set to 0x88FE, which is authorized and registered as the protocol identification number for RRP by the IEEE Registration Authority. Every frame with a value other than 0x88FE is identical to the frame in ISO/IEC 8802-3:2000, Clause 3, and is processed as a RRP fieldbus sporadic data frame. For RRP, the value shall be set according to Table 12.

Table 12 – RRP Length/Type field

Value (hexadecimal)	Meaning
0x88FE	RRP-PDU

8.3 Encoding of RRP_FrameHDR

8.3.1 Version and length

This field indicates the protocol version and the length for the RRP protocol. It shall be coded as data type Unsigned16, and the individual bits shall have the following meaning:

Bit 0 – 10: length
The value indicates the number of octets of the frame including the FCS field.

Bit 11 – 15: version
The version is represented by two bits for the major version and three bits for the minor version. This field shall be coded with the values according to Table 13.

Table 13 – Version

Field Name	Position	Meaning
Major	Bit 14 – 15	RRP protocol major version
Minor	Bit 11 – 13	RRP protocol minor version

8.3.2 DST_addr

8.3.2.1 General

This field indicates the destination RRP device identifier of the node to which the frame is sent. It shall be coded as data type Unsigned16 and set according to Table 14.

Table 14 – DST_addr

Value (hexadecimal)	Meaning
0xFFFF	broadcast address
0xFFFE	network control address (C_NCM_ADDR)
0xFFFD – 0xFFDE	user-defined multicast address
0xFFDD	invalid address
0x0100 – 0xFFDC	reserved
0x0000 – 0x00FF	regular RRP device address

8.3.2.2 Broadcast address

If the destination RRP device identifier is 0xFFFF, the destination MAC address field contains the ISO/IEC 8802-3:2000 broadcast MAC address.

8.3.2.3 Network control address

The RRP protocol defines a special MAC address, 00-E0-91-02-05-99 (NCM_MAC_ADDR), for sharing network management information. Every message received through the NCM_MAC_ADDR updates the network management information.

If the destination RRP device address is 0xFFFE (NCM_ADDR), the destination MAC address field contains NCM_MAC_ADDR. However, all NCM messages except NCM_ACK_RNMS and NCM_CHECK_RNMS are transmitted using NCM_ADDR as the destination RRP device address.

8.3.2.4 User-defined multicast address

A user-defined multicast address is used to indicate multiple recipients. However, user-defined multicast addressing is not a mandatory feature in this standard. It is designed for use in a special application system that requires multicast communication. Therefore, user-defined multicast addressing is not interoperable between heterogeneous devices. The destination RRP device address range 0xFFFD-0xFFDE is used to specify the user-defined multicast address. However, the method of using the user-defined multicast address is not specified in this standard and is considered a local responsibility. This specification does not restrict the use of user-defined multicast addresses, nor is it a mandatory feature.

8.3.3 SRC_addr

This field indicates the source RRP device address of the node that generates the frame. It shall be coded as data type Unsigned16 and set according to Table 15.

Table 15 – SRC_addr

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	source RRP device address

8.3.4 Frame Control (FC)

8.3.4.1 General

This field indicates the frame control information. It shall be coded as data type Unsigned16. It is a bit set encoded as follows:

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td colspan="8">NCMT</td><td colspan="4">ToS</td><td colspan="2">PRI</td><td>RES</td><td>VoE</td></tr>
</table>

Bit 0 — 7: Network Control Message Type (NCMT)

This field shall be coded with the values according to Table 16.

Table 16 – Network control message type

Value (hexadecimal)	Meaning	Usage
0x00	Reserved	—
0x01	NCM_FAMILY_REQ	Mandatory
0x02	NCM_FAMILY_RES	Mandatory
0x03	NCM_MEDIA_LINKED	Mandatory
0x04	NCM_ADV_THIS	Mandatory
0x05	NCM_LINE_START	Mandatory
0x06	NCM_RING_START	Mandatory
0x07	NCM_ACK_RNMS	Mandatory
0x08	NCM_CHECK_RNMS	Mandatory
0x09-0xFF	Reserved	—

Bit 8 — 11: Type of Service (ToS)

This field shall be coded with the values according to Table 17.

Table 17 – Type of service

Value (hexadecimal)	Meaning	Usage
0x00	Network Control Message (NCM)	Mandatory
0x01	unconfirmed service request	Optional
0x02-0x0F	Reserved	—

Bit 12 — 13: Priority (PRI)

This field shall be coded with the values according to Table 18.

Table 18 – Priority

Value (hexadecimal)	Meaning
0x00	lowest priority
0x01-0x02	…
0x03	highest priority

Bit 14: Reserved (RES)
This field shall be set to 0.

Bit 15: Validation of Extension code (VoE)
This field shall be coded with the values according to Table 19.

Table 19 – Validation of extension code

Value (decimal)	Meaning
0	EXT Code is invalid
1	EXT Code is valid

8.3.4.2 Network control message type

This field indicates the type of network control message. The individual values shall have the following meanings:

0x01: NCM_FAMILY_REQ
This value is used to ask the device newly connected through an R-port if it is a RRP device. This message is transmitted through the R-port that generated the PHY link-up events. This message shall be not forwarded to the other port.

0x02: NCM_FAMILY_RES
This value is used to confirm whether the recipient is a RRP device when the recipient receives the NCM_FAMILY_REQ message from the newly linked device. This message is transmitted through the R-port used to receive the NCM_FAMILY_REQ message. This message shall be not forwarded to the other port.

0x03: NCM_MEDIA_LINKED
This value is used to indicate that a new RRP link has been established through the R-port. This message is transmitted through the newly activated R-port. When the recipient receives this message, the recipient increments the hop count in the frame and forwards the frame through the other R-port. This message is discarded by the LNM or the device that generated the message.

0x04: NCM_ADV_THIS
This value is used to transmit the recipient’s local device information when the recipient receives the NCM_MEDIA_LINKED message from the newly linked device. This message is transmitted through the R-port used to receive the NCM_MEDIA_LINKED message.

0x05: NCM_LINE_START
This value is used to advise that the network topology has been automatically configured as a line network. This message is initiated by the RRP protocol machine whose state changed to LNM.

0x06: NCM_RING_START
This value is used to advise that the network topology has been automatically configured as a ring network. This message is initiated and transmitted through both R-ports by the RRP protocol machine the state of which changed to RNMP.

0x07: NCM_ACK_RNMS
This value is used by the RNMS device to advise that the RNMS has been successfully selected. The NCM_ACK_RNMS message is transmitted from the RNMS to the RNMP through the received R-port of NCM_RING_START or NCM_CHECK_RNMS message.

0x08: NCM_CHECK_RNMS
This value is used to request the NCM_ACK_RNMS message from the RNMS device in the case where the RNMP device wants to check the network integrity. The NCM_CHECK_RNMS message is transmitted from the RNMP to the RNMS with the address of RNMS through both of R-port1 and R-port2.

NOTE All the message of RRP network with a specific destination address is using one R-port for the shortest path for efficiency. But the NCM_CHECK_RNMS message is also used to check integrity of RRP network, NCM_CHECK_RNMS message is transmitted through both of R-port1 and R-port2.

8.3.4.3 Type of Service (ToS)

This field indicates the type of data link (DL) service. A value of 0x00 indicates a network control message among RRP devices, and 0x01 indicates an unconfirmed service request among users.

8.3.4.4 Priority (PRI)

This field indicates the frame priority. It contains the value of the message priority parameter for the RRP service. The highest priority is 0x03 and the lowest is 0x00.

8.3.4.5 Validation of extension code (VoE)

If the frame has an extension field, VoE is set to TRUE; otherwise VoE is set to FALSE.

8.4 Encoding of data and pad

8.4.1 General

This field indicates the data field received from a user. Network control messages are used to transfer network control messages among RRP devices. Eight message types are provided to share the network information.

8.4.2 Encoding of FamilyReq

The FamilyReq frame is encoded as specified in Table 20.

Table 20 – Encoding of FamilyReq frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information

No.	Parameter name	Data type	Octet offset	Octet length	Description
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.

8.4.3 Encoding of FamilyRes

The FamilyRes frame is encoded as specified in Table 21.

Table 21 – Encoding of FamilyRes frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.

8.4.4 Encoding of MediaLinked

The MediaLinked frame is encoded as specified in Table 22.

Table 22 – Encoding of MediaLinked frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1

No.	Parameter name	Data type	Octet offset	Octet length	Description
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.

8.4.5 Encoding of AdvThis

The AdvThis frame is encoded as specified in Table 23.

Table 23 – Encoding of AdvThis frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.

8.4.6 Encoding of LineStart

The LineStart frame is encoded as specified in Table 24

Table 24 – Encoding of LineStart frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags

No.	Parameter name	Data type	Octet offset	Octet length	Description
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.
16	Topology	Unsigned8	64	1	RRP network topology
17	Collision count	Unsigned8	65	1	Device address collision count between remote devices
18	Device count	Unsigned16	66	2	Device count for the network segment
19	Topology change count	Unsigned16	68	2	Network topology change count
20	Network flags	Unsigned16	70	2	Network event flags
21	Last topology change time	TIMEOFDAY	72	6	Date and time when the network topology last was changed
22	Reserved2	Unsigned16	78	2	Reserved, – set to 0.
23	RNMP device UID	UniqueDeviceID64	80	8	UID of the RNMP device
24	RNMS device UID	UniqueDeviceID64	88	8	UID of the RNMS device
25	LNM device UID for R-port1	UniqueDeviceID64	96	8	UID of the LNM device in R-port1 direction
26	LNM device UID for R-port2	UniqueDeviceID64	104	8	UID of the LNM device in R-port2 direction

8.4.7 Encoding of RingStart

The RingStart frame is encoded as specified in Table 25.

Table 25 – Encoding of RingStart frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1

No.	Parameter name	Data type	Octet offset	Octet length	Description
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.
16	Topology	Unsigned8	64	1	RRP network topology
17	Collision count	Unsigned8	65	1	Device address collision count between remote devices
18	Device count	Unsigned16	66	2	Device count for the network segment
19	Topology change count	Unsigned16	68	2	Network topology change count
20	Network flags	Unsigned16	70	2	Network event flags
21	Last topology change time	TIMEOFDAY	72	6	Date and time when the network topology last was changed
22	Reserved2	Unsigned16	78	2	Reserved, – set to 0.
23	RNMP device UID	UniqueDeviceID64	80	8	UID of the RNMP device
24	RNMS device UID	UniqueDeviceID64	88	8	UID of the RNMS device
25	LNM device UID for R-port1	UniqueDeviceID64	96	8	UID of the LNM device in the R-port1 direction
26	LNM device UID for R-port2	UniqueDeviceID64	104	8	UID of the LNM device in the R-port2 direction

8.4.8 Encoding of AckRNMS

The AckRNMS frame is encoded as specified in Table 26.

Table 26 – Encoding of AckRNMS frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information

No.	Parameter name	Data type	Octet offset	Octet length	Description
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.

8.4.9 Encoding of CheckRNMS

The CheckRNMS frame is encoded as specified in Table 27.

Table 27 – Encoding of CheckRNMS frame

No.	Parameter name	Data type	Octet offset	Octet length	Description
1	Device address	Unsigned16	0	2	Local device address
2	Device flags	Unsigned16	2	2	Local device flags
3	Device type	Unsigned16	4	2	Local device type
4	Hop count	Unsigned16	6	2	Hop count
5	Device UID	UniqueDeviceID64	8	8	Local device unique ID
6	Device UID for R-port1	UniqueDeviceID64	16	8	Unique ID of device connected through R-port1
7	Device UID for R-port2	UniqueDeviceID64	24	8	Unique ID of device connected through R-port2
8	MAC address	Unsigned48	32	6	Local device MAC address
9	Reserved0	Unsigned16	38	2	Reserved, – set to 0.
10	R-port1 information	Unsigned8	40	1	Local device R-port1 information
11	R-port2 information	Unsigned8	41	1	Local device R-port2 information
12	Device state	Unsigned8	42	1	Local device state
13	Protocol version	Unsigned8	43	1	Local device protocol version
14	Device description	VisibleString	44	16	Device description string
15	Reserved1	Unsigned32	60	4	Reserved, – set to 0.

8.5 Frame Check Sequence (FCS)

The FCS construction, polynomial, and expected residual are identical to those in ISO/IEC 8802-3:2000, Clause 3.

9 RRP protocol machine

9.1 Protocol state machine description

The RRP protocol machine is shown in Figure 19.

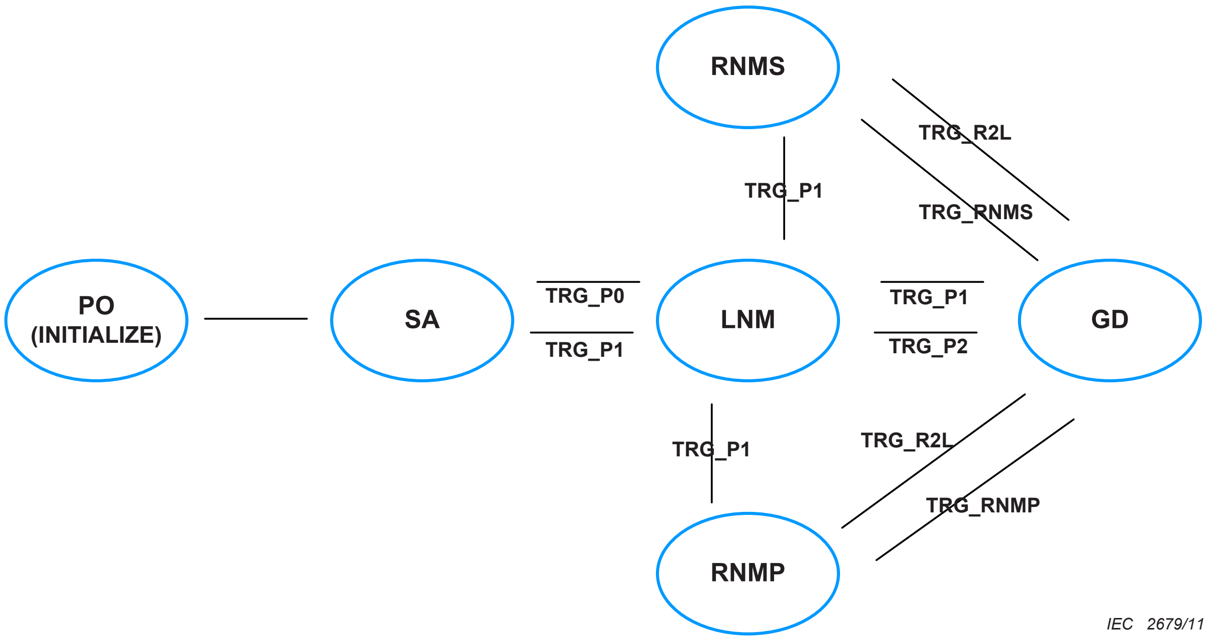

Figure 19 – RRP protocol state machine

The text below is an explanation of the overall actions performed in the states. If there is a difference in interpretation between this text and the state machine, then the state machine governs. A RRP device has six states: PO, SA, LNM, GD, RNMP, and RNMS.

PO (power on, initialize)
This state means that the local device is starting up its power supply and running a self initialization procedure. After initial process is completed, the device changes its state to SA.

SA (standalone)
This state means that the local initialization procedure has been successfully completed and the device is ready to be linked to the other devices. In the SA state, the RRP device tries to find the other devices on the network. When a link is established, the state changes to the LNM state.

LNM (line network manager)
This state means that the local device is located at the end of a line network. The LNM device is linked to the line network through one of its two R-ports. Both frame forward functions are disabled in the LNM device.

GD (general device)
This state means that the local device is connected to the network through both its R-ports. Both frame forward functions are enabled in the GD device. However, the frame forward functions in the GD device are suspended until the device receives a LineStart message (NCM_LINE_START) or a RingStart message (NCM_RING_START).

RNMP (primary ring network manager)
This state means that the local device has been automatically selected as the primary ring network manager in a ring network. The RNMP device selects one of its neighbouring devices as the secondary ring network manager (RNMS) using the RingStart message (NCM_RING_START). The RNMP device disables the frame forward function in the RNMS direction but keeps the frame forward function in the other direction enabled.

RNMS (secondary ring network manager)
This state means that the local device is selected as the secondary ring network manager in a ring network. The RNMS device disables the frame forward function in the RNMP direction but keeps the frame forward function in the other direction enabled.

Each state has transactions. These transactions are defined as follows:

TRG_P0
This transaction indicates that both ports of the device are LINK_DOWN or not acknowledged LINK_UP.

TRG_P1
This transaction indicates that only one port of the device is LINK_UP and LINK_UP has been acknowledged.

TRG_P2
This transaction indicates that both ports of the device are LINK_UP and LINK_UP has been acknowledged.

TRG_R2L
This transaction indicates that the device has received NCM_LINE_START message.

TRG_RNMP
This transaction indicates that the GD has elected as a RNMP.

TRG_RNMS
This transaction indicates that the GD has selected as a RNMS by the RNMP.

9.2 Local parameters and variables for protocol state

9.2.1 General

Local parameters and variables include local device information, network information, and path table information. Local data link information, such as the device address and the status of each R-port, is stored in the local device information. The network topology and the network-related variables are stored in the network information, and the device profile and path information of the other devices on the network are summarized in the path table.

9.2.2 Variables to support local device information management

To maintain the network topology, each device manages a device database that includes the local device information and information about other devices. Table 28 shows a list of device information management variables.

Table 28 – Variables to support device information management

Name	Data type	Description
DEV_ADDR	Unsigned16	Local device address
DEV_FLAG	Unsigned16	Local device flags
DEV_STATE	Unsigned8	Local device state
DEV_UID	UniqueDeviceID64	Local device unique ID
DEV_UID_RP1	UniqueDeviceID64	Unique ID of device connected through R-port1
DEV_UID_RP2	UniqueDeviceID64	Unique ID of device connected through R-port2
MAC_ADDR	Unsigned48	Local device MAC address
PORT1_INFO	Unsigned8	Local device R-port1 information
PORT2_INFO	Unsigned8	Local device R-port2 information
PROTOCOL_VER	Unsigned8	Local device protocol version
DEV_TYPE	Unsigned16	Local device type
DEV_DESC	VisibleString[16]	Device description string
HOP_CNT	Unsigned16	Hop count

9.2.3 Variables to support network information management

The network information is managed automatically by the RRP protocol machine. The network information variables are summarized in Table 29.

Table 29 – Variables to support managing network information

Name	Data type	Description
RRP_NET_TPG	Unsigned8	RRP network topology
UID_RNMP	UniqueDeviceID64	UID of the RNMP device
UID_RNMS	UniqueDeviceID64	UID of the RNMS device
UID_LNM_RP1	UniqueDeviceID64	UID of the LNM device in R-port1 direction
UID_LNM_RP2	UniqueDeviceID64	UID of the LNM device in R-port2 direction

9.2.4 Variables to support device path information management

The path table is managed by the RRP protocol machine. The variables for a path table item are defined in Table 30.

Table 30 – Variables to support device path information management

Name	Data type	Description
path-DEV_ADDR	Unsigned16	Peer device address
path-HOP_CNT_RP1	Unsigned16	Hop count in R-port1 direction
path-HOP_CNT_RP2	Unsigned16	Hop count in R-port2 direction
path-PREFER_RP	Unsigned8	R-port with the smaller hop count value to the peer device
path-DST_RP	Unsigned8	Selected R-port for sending a frame to the destination device address
path-DEV_STATE	Unsigned8	Peer device state
path-MAC_ADDR	Unsigned48	Peer device ISO/IEC 8802-3:2000 MAC address
path-PORT1_INFO	Unsigned8	Peer device local R-port1 information
path-PORT2_INFO	Unsigned8	Peer device local R-port2 information
path-PROTOCOL_VER	Unsigned8	Peer device protocol version
path-DEV_TYPE	Unsigned16	Peer device application device type
path-DEV_DESC	VisibleString[16]	Peer device description
path-DEV_UID	UniqueDeviceID64	Peer device UID
path-DEV_UID_RP1	UniqueDeviceID64	Device UID of the device connected through the peer device R-port1
path-DEV_UID_RP2	UniqueDeviceID64	Device UID of the device connected through the peer device R-port2

9.2.5 Variables of Received RRP Frame

The variables of the Received RRP frame are listed in Table 31.

Table 31 – Variables of Received RRP Frame

Name	Data type	Description
rcv-DEV_ADDR	Unsigned16	Device address in received frame
rcv-DEV_STATE	Unsigned8	Device state in received frame

Name	Data type	Description
rcv-DEV_UID	UniqueDeviceID64	Unique ID of device in received frame
rcv-PORT1_INFO	Unsigned8	R-port1 information in received frame
rcv-PORT2_INFO	Unsigned8	R-port2 information in received frame
rcv-HOP_CNT	Unsigned16	Hop count in received frame
rcv-UID_RNMP	UniqueDeviceID64	UID of the RNMP device in received frame
rcv-UID_RNMS	UniqueDeviceID64	UID of the RNMS device in received frame

9.2.6 Local variables for protocol state

The local variables for the protocol state are listed in Table 32.

Table 32 – Local variables for protocol state

Name	Data type	Description
Link_status	Boolean	PHY link status of local device
RRP_FamilyReqT	Unsigned32	FamilyRes frame waiting time Default value is 3 ms
RRP_MediaLinkedT	Unsigned32	AdvThis frame waiting time Default value is 3 ms
RRP_AckRNMST	Unsigned32	AckRNMS frame waiting time Default value is 3 ms
RRP_ChkRNMST	Unsigned32	ChkRNMS frame waiting time Default value is 3 ms
RRP_ChangeRingStateT	Unsigned32	Ring state change timeout Default value is 3 ms
RRP_RetryCount	Unsigned32	RRP network integrity check count Default value is 3 ms
RRP_ChkIntegrityT	Unsigned32	ChkRNMS time interval time Default value is 3 ms

9.2.7 Constants for protocol state

The constants for the protocol state are listed in Table 33.

Table 33 – Constants for protocol state

Constant	Description	Value
PHY_LINK_UP	Event generated when PHY link is up	TRUE
PHY_LINK_DOWN	Event generated when PHY link is down	FALSE
TRUE	Return value 1	1
FALSE	Return value 0	0
INVALID_UID	Invalid unique identification	0
INVALID_R_PORT	Invalid R-port state	0
PORT_LINK_DOWN	R-port link down	0x01
PORT_CFM_FAMILY	Received Family_Frame.Cnf from R-port	0x02
PORT_WAIT_ADV	Waiting for AdvThis_Frame.Cnf from R-port	0x04
PORT_WAIT_ML	Waiting for MediaLinked_Frame.Ind from R-port	0x08
PORT_CFM	Device state confirm	0x10

9.3 State transitions

The state transitions of the RRP protocol state machine are specified in Table 34.

Table 34 – RRP State transitions

No.	Current state	Event /Condition =>Action	Next state
1	PO	POWER-ON or RESET / => INIT_DEV_INFO( ) INIT_NET_INFO( ) INIT_PATH_INFO( ) MAC-RESET.req{ } Ph-RESET.req{ } Set_Block_Port(INVALID_R_PORT) Clear_Port_Info(R-port1) Clear_Port_Info(R-port2) DEV_STATE := SA STORE_PATH_INFO(DEV_ADDR)	SA
2	SA	Phy_Link_Change.Ind(R-port, Link_status) /R-port == R-port1 && Link_status == PHY_LINK_UP => Set_Block_Port(R-port2) Start_Timer(RRP_FamilyReqT) Family_Frame.Req(R-port)	SA
3	SA	Phy_Link_Change.Ind(R-port, Link_status) /R-port == R-port2 && Link_status == PHY_LINK_UP => Set_Block_Port(R-port1) Start_Timer(RRP_FamilyReqT) Family_Frame.Req(R-port)	SA
4	SA	Family_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port)	SA
5	SA	Timer(RRP_FamilyReqT) expired / => Start_Timer(RRP_FamilyReqT) Family_Frame.Req(R-port)	SA

No.	Current state	Event /Condition =>Action	Next state
6	SA	Family_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => Stop_Timer(RRP_FamilyReqT) Set_Neighbor_UID(R-port, rcv-DEV_UID) Set_Port_Info(R-port, PORT_CFM_FAMILY) STORE_PATH_INFO(rcv-DEV_ADDR) Start_Timer(RRP_MediaLinkedT) MediaLinked_Frame.Req(R-port)	SA
7	SA	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_CFM_FAMILY) == FALSE => Ignore	SA
8	SA	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ML) == TRUE => Set_Port_Info(R-port, PORT_CFM) Set_Block_Port(INVALID_R_PORT) STORE_PATH_INFO(rcv-DEV_ADDR) RRP_NET_TPG := NET_TPG_LINE Set_LNM_UID(R-port, DEV_UID) DEV_STATE := LNM AdvThis_Frame.Res(R-port) RRP_Line_Start_Frame.Req(R-port)	LNM
9	SA	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ML) == FALSE => Set_Port_Info(R-port, PORT_WAIT_ADV) STORE_PATH_INFO(rcv-DEV_ADDR) AdvThis_Frame.Res(R-port)	SA
10	SA	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != Get_Neighbor_UID(R-port) => STORE_PATH_INFO(rcv-DEV_ADDR) AdvThis_Frame.Res(R-port)	SA

No.	Current state	Event /Condition =>Action	Next state
11	SA	Timer(RRP_MediaLinkedT) expired / => Start_Timer(RRP_MediaLinkedT) MediaLinked_Frame.Req(R-port)	SA
12	SA	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == FALSE => Ignore	SA
13	SA	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ADV) == TRUE && (rcv-PORT1_INFO == PORT_LINK_DOWN || rcv-PORT2_INFO == PORT_LINK_DOWN) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_CFM) Set_Block_Port(INVALID_R_PORT) STORE_PATH_INFO(rcv-DEV_ADDR) Set_LNM_UID(R-port, DEV_UID) DEV_STATE := LNM RRP_NET_TPG := NET_TPG_LINE RRP_Line_Start_Frame.Req(R-port)	LNM
14	SA	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ADV) == FALSE && (rcv-PORT1_INFO == PORT_LINK_DOWN || rcv-PORT2_INFO == PORT_LINK_DOWN) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_WAIT_ML) STORE_PATH_INFO(rcv-DEV_ADDR)	SA
15	SA	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && rcv-PORT1_INFO != PORT_LINK_DOWN && rcv-PORT2_INFO != PORT_LINK_DOWN => STORE_PATH_INFO(rcv-DEV_ADDR)	SA

No.	Current state	Event /Condition =>Action	Next state
16	SA	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN => Set_Block_Port(INVALID_R_PORT) Clear_Port_Info(R-port1) Clear_Port_Info(R-port2) INIT_PATH_INFO( ) STORE_PATH_INFO(DEV_ADDR)	SA
17	LNM	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_UP => Start_Timer(RRP_FamilyReqT) Family_Frame.Req(R-port)	LNM
18	LNM	Family_Frame.Ind (R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port)	LNM
19	LNM	Timer(RRP_FamilyReqT) expired / => Start_Timer(RRP_FamilyReqT) Family_Frame.Req(R-port)	LNM
20	LNM	Family_Frame.Cnf (R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => Stop_Timer(RRP_FamilyReqT) Set_Neighbor_UID(R-port, rcv-DEV_UID) Set_Port_Info(R-port, PORT_CFM_FAMILY) STORE_PATH_INFO(rcv-DEV_ADDR) Start_Timer(RRP_MediaLinkedT) MediaLinked_Frame.Req(R-port)	LNM
21	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_CFM_FAMILY) == FALSE => Ignore	LNM

No.	Current state	Event /Condition =>Action	Next state
22	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ML) == TRUE => Set_Port_Info(R-port, PORT_CFM) DEV_STATE := GD STORE_PATH_INFO(rcv-DEV_ADDR) Clear_LNM_UID(R-port) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port)	GD
23	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ML) == FALSE => Set_Port_Info(R-port, PORT_WAIT_ADV) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port)	LNM
24	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != Get_Neighbor_UID(R-port) && rcv-DEV_UID != DEV_UID => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port)	LNM
25	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == DEV_UID => Ignore	LNM
26	LNM	Timer(RRP_MediaLinkedT) expired / => Start_Timer(RRP_MediaLinkedT) MediaLinked_Frame.Req(R-port)	LNM
27	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == FALSE => Ignore	LNM

No.	Current state	Event /Condition =>Action	Next state
28	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ADV) == TRUE && (rcv-DEV_UID == UID_LNM_RP1 || rcv-DEV_UID == UID_LNM_RP2) && rcv-DEV_UID == Get_Neighbor_UID(R-port) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_CFM) DEV_STATE := GD STORE_PATH_INFO(rcv-DEV_ADDR) Clear_LNM_UID(R-port) Forward_Frame(R-port, AdvThis_Frame.Cnf)	GD
29	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ADV) == TRUE && (rcv-DEV_UID != UID_LNM_RP1 && rcv-DEV_UID != UID_LNM_RP2) && (rcv-PORT1_INFO == PORT_LINK_DOWN || rcv-PORT2_INFO == PORT_LINK_DOWN) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_CFM) DEV_STATE := GD STORE_PATH_INFO(rcv-DEV_ADDR) Clear_LNM_UID(R-port) Forward_Frame(R-port, AdvThis_Frame.Cnf)	GD
30	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ADV) == FALSE && (rcv-DEV_UID == UID_LNM_RP1 || rcv-DEV_UID == UID_LNM_RP2) && rcv-DEV_UID == Get_Neighbor_UID(R-port) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_WAIT_ML) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Cnf)	LNM

No.	Current state	Event /Condition =>Action	Next state
31	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_WAIT_ADV) == FALSE && (rcv-DEV_UID != UID_LNM_RP1 && rcv-DEV_UID != UID_LNM_RP2) && (rcv-PORT1_INFO == PORT_LINK_DOWN || rcv-PORT2_INFO == PORT_LINK_DOWN) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_WAIT_ML) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Cnf)	LNM
32	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && (rcv-DEV_UID == UID_LNM_RP1 || rcv-DEV_UID == UID_LNM_RP2) && rcv-DEV_UID != Get_Neighbor_UID(R-port) && rcv-DEV_UID != DEV_UID => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Cnf)	LNM
33	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && (rcv-DEV_UID != UID_LNM_RP1 && rcv-DEV_UID != UID_LNM_RP2) && (rcv-PORT1_INFO != PORT_LINK_DOWN && rcv-PORT2_INFO != PORT_LINK_DOWN) && rcv-DEV_UID != DEV_UID => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Cnf)	LNM
34	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && rcv-DEV_UID == DEV_UID => RRP_NET_TPG := NET_TPG_RING Start_Timer(RRP_ChangeRingStateT)	LNM
35	LNM	Timer(RRP_ChangeRingStateT) expired / => Start_Timer(RRP_ChangeRingStateT)	LNM

No.	Current state	Event /Condition =>Action	Next state
36	LNM	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && Chk_Port_Info(R-port, PORT_CFM) == FALSE && R-port == R-port1 => Clear_Port_Info(R-port) DELETE_PATH_INFO(R-port) RRP_Line_Start_Frame.Req(R-port2)	LNM
37	LNM	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && Chk_Port_Info(R-port, PORT_CFM) == FALSE && R-port == R-port2 => Clear_Port_Info(R-port) DELETE_PATH_INFO(R-port) RRP_Line_Start_Frame.Req(R-port1)	LNM
38	LNM	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && Chk_Port_Info(R-port, PORT_CFM) == TRUE => INIT_DEV_INFO( ) INIT_NET_INFO( ) INIT_PATH_INFO( ) Set_Block_Port(INVALID_R_PORT) Clear_Port_Info(R-port1) Clear_Port_Info(R-port2) DEV_STATE := SA STORE_PATH_INFO(DEV_ADDR) MAC-RESET.req{ } Ph-RESET.req{ }	SA
39	LNM	RRP_Line_Start_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => Store_LNM_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR)	LNM
40	GD	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == DEV_UID => Ignore	GD

No.	Current state	Event /Condition =>Action	Next state
41	GD	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != DEV_UID => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port)	GD
42	GD	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /rcv-DEV_UID == DEV_UID => RRP_NET_TPG := NET_TPG_RING Start_Timer(RRP_ChangeRingStateT)	GD
43	GD	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO) /rcv-DEV_UID != DEV_UID => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Cnf)	GD
44	GD	Timer(RRP_ChangeRingStateT) expired /RRP_NET_TPG == NET_TPG_RING && CheckRNMP(UID_RNMP, UID_RNMS) == TRUE => Stop_Timer(RRP_ChangeRingStateT) DEV_STATE := RNMP Clear_LNM_UID(0) RRP_Ring_Start_Frame.Req(R-port2, UID_RNMP, UID_RNMS) Start_Timer(RRP_AckRNMST)	RNMP
45	GD	Timer(RRP_ChangeRingStateT) expired /RRP_NET_TPG == NET_TPG_RING || CheckRNMP(UID_RNMP, UID_RNMS) == FALSE => Stop_Timer(RRP_ChangeRingStateT)	GD
46	GD	RRP_Ring_Start_Frame.Ind(R-port, rcv-DEV_ADDR, rcv-UID_RNMP, rcv-UID_RNMS) /rcv-UID_RNMS == DEV_UID => DEV_STATE := RNMS Clear_LNM_UID(0) UID_RNMP := rcv-UID_RNMP UID_RNMS := rcv-UID_RNMS STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, RRP_Ring_Start_Frame.Ind) AckRNMS_Frame.Res(R-port)	RNMS

No.	Current state	Event /Condition =>Action	Next state
47	GD	RRP_Ring_Start_Frame.Ind(R-port, rcv-DEV_ADDR, rcv-UID_RNMP, rcv-UID_RNMS) /rcv-UID_RNMS != DEV_UID => Clear_LNM_UID(0) UID_RNMP := rcv-UID_RNMP UID_RNMS := rcv-UID_RNMS STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, RRP_Ring_Start_Frame.Ind)	GD
48	GD	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && R-port == R-port1 => Clear_Port_Info(R-port) DEV_STATE := LNM DELETE_PATH_INFO(R-port) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, DEV_UID) RRP_Line_Start_Frame.Req(R-port2)	LNM
49	GD	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && R-port == R-port2 => Clear_Port_Info(R-port) DEV_STATE := LNM DELETE_PATH_INFO(R-port) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, DEV_UID) RRP_Line_Start_Frame.Req(R-port1)	LNM
50	GD	RRP_Line_Start_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => STORE_PATH_INFO(rcv-DEV_ADDR) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, rcv-DEV_UID) Forward_Frame(R-port, RRP_Line_Start_Frame.Ind)	GD

No.	Current state	Event /Condition =>Action	Next state
51	RNMP	AckRNMS_Frame.Cnf(R-port, rcv-DEV_UID) /rcv-DEV_UID == UID_RNMS => Stop_Timer(RRP_AckRNMST)	RNMP
52	RNMP	AckRNMS_Frame.Cnf(R-port, rcv-DEV_UID) /rcv-DEV_UID != UID_RNMS => ignore	RNMP
53	RNMP	Timer(RRP_AckRNMST) expired / => CheckRNMS_Frame.Req(UID_RNMS) Start_Timer(RRP_AckRNMST)	RNMP
54	RNMP	RRP_Ring_Start_Frame.Ind(R-port, rcv-DEV_ADDR, rcv-UID_RNMP, rcv-UID_RNMS) /rcv-UID_RNMP == DEV_UID => Ignore	RNMP
55	RNMP	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && R-port == R-port1 => Clear_Port_Info(R-port) DEV_STATE := LNM DELETE_PATH_INFO(R-port) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, DEV_UID) RRP_Line_Start_Frame.Req(R-port2)	LNM
56	RNMP	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && R-port == R-port2 => Clear_Port_Info(R-port) DEV_STATE := LNM DELETE_PATH_INFO(R-port) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, DEV_UID) RRP_Line_Start_Frame.Req(R-port1)	LNM

No.	Current state	Event /Condition =>Action	Next state
57	RNMP	RRP_Line_Start_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => DEV_STATE := GD STORE_PATH_INFO(rcv-DEV_ADDR) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, rcv-DEV_UID) Forward_Frame(R-port, RRP_Line_Start_Frame.Ind)	GD
58	RNMS	RRP_Ring_Start_Frame.Ind(R-port, rcv-DEV_ADDR, rcv-UID_RNMP, rcv-UID_RNMS) /rcv-UID_RNMS == DEV_UID => Clear_LNM_UID(0) UID_RNMP := rcv-UID_RNMP UID_RNMS := rcv-UID_RNMS STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, RRP_Ring_Start_Frame.Ind) AckRNMS_Frame.Res(R-port)	RNMS
59	RNMS	CheckRNMS_Frame.Ind(R-port, rcv-DEV_ADDR) / => STORE_PATH_INFO(rcv-DEV_ADDR) AckRNMS_Frame.Res(R-port)	RNMS
60	RNMS	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && R-port == R-port1 => Clear_Port_Info(R-port) DEV_STATE := LNM DELETE_PATH_INFO(R-port) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, DEV_UID) RRP_Line_Start_Frame.Req(R-port2)	LNM

No.	Current state	Event /Condition =>Action	Next state
61	RNMS	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && R-port == R-port2 => Clear_Port_Info(R-port) DEV_STATE := LNM DELETE_PATH_INFO(R-port) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, DEV_UID) RRP_Line_Start_Frame.Req(R-port1)	LNM
62	RNMS	RRP_Line_Start_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / => DEV_STATE := GD STORE_PATH_INFO(rcv-DEV_ADDR) UID_RNMP := INVALID_UID UID_RNMS := INVALID_UID RRP_NET_TPG := NET_TPG_LINE Store_LNM_UID(R-port, rcv-DEV_UID) Forward_Frame(R-port, RRP_Line_Start_Frame.Ind)	GD

9.4 Function descriptions

The RRP functions shall be according to Table 35.

Table 35 – RRP Function descriptions

Function name	Operations
Phy_Link_Change.Ind(R-port, Link_status)	Receive a local link change indication of the R-port from PHY R-port := port that caused the local link change indication. Link_status := PHY_LINK_UP or PHY_LINK_DOWN (depends on the local link change indication)
Family_Frame.Req(R-port)	Create and send RRP Family Request frame Destination MAC Address := NCM_MAC_ADDR DST_addr := C_NCM_ADDR SRC_addr := DEV_ADDR FC.NCMT := NCM_FAMILY_REQ RRP-DATA := Local_Device_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
Family_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR)	Receive RRP Family Request frame R-port := port that received the RRP Family Request frame rcv-DEV_UID := the DEV_UID of the received frame rcv-DEV_ADDR := the DEV_ADDR of the received frame
Family_Frame.Res(R-port)	Create and send RRP Family Response frame Destination MAC Address := NCM_MAC_ADDR DST_addr := C_NCM_ADDR SRC_addr := DEV_ADDR FC.NCMT := NCM_FAMILY_RES RRP-DATA := Local_Device_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
Family_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR)	Receive RRP Family Response frame R-port := port that received the RRP Family Response frame rcv-DEV_UID := the DEV_UID of the received frame rcv-DEV_ADDR := the DEV_ADDR of the received frame

Function name	Operations
MediaLinked_Frame.Req(R-port)	Create and send RRP Media Linked frame Destination MAC Address := NCM_MAC_ADDR DST_addr := C_NCM_ADDR SRC_addr := DEV_ADDR FC.NCMT := NCM_MEDIA_LINKED RRP-DATA := Local_Device_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR)	Receive RRP Media Linked frame R-port := port that received the RRP Media Linked frame rcv-DEV_UID := the DEV_UID of the received frame rcv-DEV_ADDR := the DEV_ADDR of the received frame
AdvThis_Frame.Res(R-port)	Create and send RRP Adv This frame Destination MAC Address := NCM_MAC_ADDR DST_addr := C_NCM_ADDR SRC_addr := DEV_ADDR FC.NCMT := NCM_ADV_THIS RRP-DATA := Local_Device_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR, rcv-PORT1_INFO, rcv-PORT2_INFO)	Receive RRP Adv This frame R-port := port that received the RRP Adv This frame rcv-DEV_UID := the DEV_UID of the received frame rcv-DEV_ADDR := the DEV_ADDR of the received frame rcv-PORT1_INFO := the PORT1_INFO of the received frame rcv-PORT2_INFO := the PORT2_INFO of the received frame
RRP_Line_Start_Frame.Req(R-port)	Create and send RRP Line Start frame Destination MAC Address := NCM_MAC_ADDR DST_addr := C_NCM_ADDR SRC_addr := DEV_ADDR FC.NCMT := NCM_LINE_START RRP-DATA := Local_Device_Information , Net_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
RRP_Line_Start_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR)	Receive RRP Line Start frame R-port := port that received the RRP Line Start frame rcv-DEV_UID := the DEV_UID of the received frame rcv-DEV_ADDR := the DEV_ADDR of the received frame

Function name	Operations
RRP_Ring_Start_Frame.Req(R-port, UID_RNMP, UID_RNMS)	Create and send RRP Ring Start frame Destination MAC Address := NCM_MAC_ADDR DST_addr := C_NCM_ADDR SRC_addr := DEV_ADDR FC.NCMT := NCM_RING_START RRP-DATA := Local_Device_Information, Net_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
RRP_Ring_Start_Frame.Ind(R-port, rcv-DEV_ADDR, rcv-UID_RNMP, rcv-UID_RNMS)	Receive RRP Ring Start frame R-port := port that received the RRP Ring Start frame rcv-DEV_ADDR := the DEV_ADDR of the received frame rcv-UID_RNMP := the UID_RNMP of the received frame rcv-UID_RNMS := the UID_RNMS of the received frame
AckRNMS_Frame.Res(R-port)	Create and send RRP ACK RNMS frame DST_addr := UID_RNMP SRC_addr := DEV_ADDR FC.NCMT := NCM_ACK_RNMS RRP-DATA := Local_Device_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
AckRNMS_Frame.Cnf(R-port, rcv-DEV_UID)	Receive RRP ACK RNMS frame R-port := port that received the RRP ACK RNMS frame rcv-DEV_UID := the DEV_UID of the received frame
CheckRNMS_Frame.Req(UID_RNMS)	Create and send RRP Check RNMS frame DST_addr := UID_RNMS SRC_addr := DEV_ADDR FC.NCMT := NCM_CHECK_RNMS RRP-DATA := Local_Device_Information SendFrame(R-port, RRP_HDR, RRP-DATA)
CheckRNMS_Frame.Ind(R-port, rcv-DEV_ADDR)	Receive RRP Retry RNMS frame R-port := port that received the RRP ACK RNMS frame rcv-DEV_ADDR := the DEV_ADDR of the received frame

Function name	Operations
Forward_Frame(R-port, Received_Frame)	Forward received frame from R-port R-port := port that received the frame Received_Frame := the received frame Increments the hop counts in the received frame by 1 If R-port == 1, then SendReceivedFrame(R-port2, Received_Frame) If R-port == 2, then SendReceivedFrame(R-port1, Received_Frame)
INIT_DEV_INFO()	Initialize variables to support local device information management
Set_Port_Info(R-port, Status)	Function to set the port status of R-port to Status If R-port == 1, then PORT1_INFO := (PORT1_INFO | Status) If R-port == 2, then PORT2_INFO := (PORT2_INFO | Status)
Clear_Port_Info(R-port)	Function to clear the port status of R-port If R-port == 1, then PORT1_INFO := PORT_LINK_DOWN If R-port == 2, then PORT2_INFO := PORT_LINK_DOWN
Chk_Port_Info(R-port, Status)	Function to check the port status of R-port If (Get_Port_Info(R-port) & Status) == Status, then return TRUE else return FALSE
Get_Port_Info(R-port)	Function to get the port status of R-port If R-port == 1, then return PORT1_INFO If R-port == 2, then return PORT2_INFO
Set_Neighbor_UID(R-port, rcv-DEV_UID)	Function to set the UID of the device linked through the R-port If R-port == 1, then DEV_UID_RP1 := rcv-DEV_UID If R-port == 2, then DEV_UID_RP2 := rcv-DEV_UID
Get_Neighbor_UID(R-port)	Function to get the UID of the device linked through the R-port If R-port == 1, then return DEV_UID_RP1 If R-port == 2, then return DEV_UID_RP2
INIT_NET_INFO()	Initialize variables to support network information management
Set_LNM_UID(R-port, dev_UID)	Function to set the UID of the device selected as the LNM in the R-port direction If R-port == 1, then UID_LNM_RP2 := dev_UID If R-port == 2, then UID_LNM_RP1 := dev_UID
Store_LNM_UID(R-port, dev_UID)	Function to store the UID of the device selected as the LNM in the R-port direction If R-port == 1, then UID_LNM_RP1 := dev_UID If R-port == 2, then UID_LNM_RP2 := dev_UID
Clear_LNM_UID(R-port)	Function to clear the UID of the device selected as the LNM in the R-port direction If R-port == 0, then UID_LNM_RP1 := INVALID_UID and UID_LNM_RP2 := INVALID_UID If R-port == 1, then UID_LNM_RP1 := INVALID_UID If R-port == 2, then UID_LNM_RP2 := INVALID_UID

Function name	Operations
INIT_PATH_INFO( )	Initialize variables to support device path information management
STORE_PATH_INFO(dev_ADDR)	Add or update path information using DEV_INFO in RRP-DATA from received frame If SearchPathTable(dev_ADDR) == TRUE, then update path information in path table If SearchPathTable(dev_ADDR) == FALSE, then add path information to path table
DELETE_PATH_INFO(R-port)	Function to delete path information related to R-port
CheckRNMP(UID_RNMP, UID_RNMS)	Function to check RNMP and select RNMS If the DEV_UID is the biggest UID in the path table, then UID_RNMP := DEV_UID and UID_RNMS := DEV_UID_RP1 and return TRUE If the DEV_UID is not the biggest UID in the path table, then return FALSE
Set_Block_Port(R-port)	Function to set R-port to blocked state If R-port == 0, then set R-port1 and R-port2 to active state If R-port == 1, then set R-port1 to blocked state If R-port == 2, then set R-port2 to blocked state
MAC-RESET.req{ }	Function to reset the MAC
Ph-RESET.req{ }	Function to reset the PHY
Start_Timer(t)	Function to start the timer
Stop_Timer(t)	Function to stop the timer
SendFrame(R-port, RRP_HDR, RRP-DATA)	Function to send the RRP frame R-port := port for sending a frame
SendReceivedFrame(R-port, Received_Frame)	Function to forward the RRP frame to the other R-port. R-port := port that received the frame If R-port == 1, then send the RRP frame to R-port2 If R-port == 2, then send the RRP frame to R-port1
SearchPathTable(dev_ADDR)	Function to search the path information for dev_ADDR If path information for dev_ADDR exists in the path table, then return TRUE If path information for dev_ADDR does not exist in the path table, then return FALSE

10 RRP Management Information Base (MIB)

```
-- ************************************************************************

IEC-62439-7-MIB DEFINITIONS ::= BEGIN

-- ************************************************************************
-- Imports
-- ************************************************************************

IMPORTS
       OBJECT-IDENTITY,
       OBJECT-TYPE,
       TimeTicks,
       Counter32,
       Unsigned32,
       Counter64,
       VISIBLE-STRING,
       INTEGER                   FROM SNMPv2-SMI
       Boolean                   FROM HOST-RESOURCES-MIB
       MacAddress                FROM BRIDGE-MIB
       iso                       FROM RFC1155-SMI;

-- *******************************************************************
-- Declaration of TIMEOFDAY
-- *******************************************************************
TIMEOFDAY    ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION "
        The IEC 61158-5-21 defines the structure of
        the TIMEOFDAY as a data type numeric
        identifier 12.
        "
    SYNTAX          VISIBLE STRING (SIZE (6))

-- *******************************************************************
-- Root OID
-- *******************************************************************
iec62439    MODULE-IDENTITY
            LAST-UPDATED "201306110000Z"  --  June 11, 2013
            ORGANIZATION "IEC/SC 65C"
            CONTACT-INFO "
                            International Electrotechnical Commission
                            IEC Central Office
                            3, rue de Varembe
                            P.O. Box 131
                            CH - 1211 GENEVA 20
                            Switzerland
                            Phone: +41 22 919 02 11
                            Fax: +41 22 919 03 00
                            email: info@iec.ch
                        "
            DESCRIPTION  "
                This MIB module defines the Network Management interfaces
                for the Redundancy Protocols defined by the IEC
                standard 62439.
                "

            REVISION    "201306110000Z"  --  June 11, 2013
            DESCRIPTION  "
                Consistency adjustment with other parts of IEC 62439
                "

            REVISION    "200811100000Z"  --  November 10, 2008
            DESCRIPTION  "
                Seperation of IEC 62439 into a suite of documents.
                This MIB applies to IEC 62439-7, added RRP functionality.
                "
```

```
          REVISION    "200708240000Z"  --  August 24, 2007
          DESCRIPTION  "
              Initial version of the Network Management interface for the
              Ring-based Redundancy Protocol
              "

    ::= { IEC 62439 }

-- **********************************************************************
-- Redundancy Protocols
-- **********************************************************************
mrp                 OBJECT IDENTIFIER ::= { iec62439 1 }
prp                 OBJECT IDENTIFIER ::= { iec62439 2 }
crp                 OBJECT IDENTIFIER ::= { iec62439 3 }
brp                 OBJECT IDENTIFIER ::= { iec62439 4 }
drp                 OBJECT IDENTIFIER ::= { iec62439 5 }
rrp                 OBJECT IDENTIFIER ::= { iec62439 6 }

-- **********************************************************************
-- Objects of the RRP Network Management
-- **********************************************************************

ServiceID    OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies information sufficient for local
                    identification of the RRP device that will
                    convey the service
                    "
    ::= { rrp 1 }

InvokeID    OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the invocation of the service
                    "
    ::= { rrp 2 }

DeviceAddress    OBJECT-TYPE
    SYNTAX          INTEGER
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the value for the RRP device address
                    "
    ::= { rrp 3 }

DeviceFlags    OBJECT-TYPE
    SYNTAX          BITS {
                        DeviceAddressCollision(0),
                        DeviceStateChanged(1)
                    }
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the flags for events that occurred
                    in a local device
                    "
    ::= { rrp 4 }

DeviceState    OBJECT-TYPE
    SYNTAX          INTEGER {
                        Invalid(0),
                        SA(1),
                        LNM(2),
                        GD(3),
                        RNMP(4),
                        RNMS(5)
                    }
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the device state of the RRP device
                    "
    ::= { rrp 5 }
```

```
DeviceUID    OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the unique 8-octet identification that
                    identifies a RRP device in a network
                    "
    ::= { rrp 6 }

DeviceUIDRport1    OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the UID of the device that is linked
                    through the R-port1
                    "
    ::= { rrp 7 }

DeviceUIDRport2    OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the UID of the device that is linked
                    through the R-port2
                    "
    ::= { rrp 8 }

MACAddress    OBJECT-TYPE
    SYNTAX          MacAddress
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the MAC address of the RRP device
                    "
    ::= { rrp 9 }

Rport1Information    OBJECT-TYPE
    SYNTAX          BITS {
                        PortLinkDown(0),
                        PortCFMFamily(1),
                        PortWaitADV(2),
                        PortWaitML(3)
                        PortCFM(4)
                    }
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the port information for R-port1
                    "
    ::= { rrp 10 }

Rport2Information    OBJECT-TYPE
    SYNTAX          BITS {
                        PortLinkDown(0),
                        PortCFMFamily(1),
                        PortWaitADV(2),
                        PortWaitML(3)
                        PortCFM(4)
                    }
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the port information for R-port2
                    "
    ::= { rrp 11 }

Version    OBJECT-TYPE
    SYNTAX          INTEGER
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the protocol version of the RRP device
                    "
    ::= { rrp 12 }
```

```
DeviceType    OBJECT-TYPE
    SYNTAX          INTEGER
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the local device type that represents
                    the general function of the device
                    "
    ::= { rrp 13 }

DeviceDescription    OBJECT-TYPE
    SYNTAX          VISIBLE STRING (SIZE(1..16))
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies a description of the local device
                    "
    ::= { rrp 14 }

FamilyResWaitingTime    OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the time interval betweensending the FamilyReq
                    frame andreceiving the FamilyRes frame
                    "
    ::= { rrp 15 }

AdvThisWaitingTime    OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the time interval betweensending the MediaLinked
                    frame andreceiving the AdvThis frame
                    "
    ::= { rrp 16 }

AckRNMSWaitingTime    OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the time interval betweensending the RingStart
                    frame andreceiving the AckRNMS frame
                    "
    ::= { rrp 17 }

RingStateChangeTimeout    OBJECT-TYPE
    SYNTAX          Unsigned32
    MAX-ACCESS      read-write
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the timeout to generate event for changing
                    RNMP device state
                    "
    ::= { rrp 18 }

DiagnosticInformation    OBJECT-TYPE
    SYNTAX          INTEGER {
                        NetworkInformation(1),
                        PathTableInformation(2)
                    }
    MAX-ACCESS      write-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the type of diagnostic information
                    "
    ::= { rrp 19 }
```

```
-- ********************************************************************
-- Objects of the RRP Network Information
-- ********************************************************************

NetworkTopology    OBJECT-TYPE
    SYNTAX          INTEGER {
                        Invalid(0),
                        NET_TPG_SA(1),
                        NET_TPG_LINE(2),
                        NET_TPG_RING(3)
                    }
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the type of network topology
                    "
    ::= { rrp 20 }

CollisionCnt    OBJECT-TYPE
    SYNTAX          INTEGER
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the device address collision count
                    for remote devices
                    "
    ::= { rrp 21 }

DeviceCnt    OBJECT-TYPE
    SYNTAX          INTEGER
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the total number of devices on the network
                    "
    ::= { rrp 22 }

TopologyChangeCnt    OBJECT-TYPE
    SYNTAX          INTEGER
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the topology change count
                    "
    ::= { rrp 23 }

LastTopologyChangeTime    OBJECT-TYPE
    SYNTAX          TIMEOFDAY
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the date and time at whichthe network topology
                    was last changed
                    "
    ::= { rrp 24 }

RNMPDeviceUID    OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the UID of the device selected as the RNMP
                    on the network
                    "
    ::= { rrp 25 }

RNMSDeviceUID    OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the UID of the device selected as the RNMS
                    on the network
                    "
    ::= { rrp 26 }
```

```
LNMDeviceUIDRport1    OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the UID of the device selected as the LNM
                    in the R-port1 direction
                    "
    ::= { rrp 27 }

LNMDeviceUIDRport2    OBJECT-TYPE
    SYNTAX          Counter64
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the UID of the device selected as the LNM
                    in the R-port2 direction
                    "
    ::= { rrp 28 }

NetworkFlags    OBJECT-TYPE
    SYNTAX          BITS {
                        NetworkTopologyChanged(0),
                        DeviceAddressCollisionInNetwork(1),
                        DeviceJoinedNetwork(2),
                        DeviceLeftNetwork(3)
                    }
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    specifies the flags for events that occurred
                    in the network
                    "
    ::= { rrp 29 }

-- *********************************************************************
-- Objects of the RRP Path Table Information
-- *********************************************************************

PathTableInfo    OBJECT-TYPE
    SYNTAX          SEQUENCE OF PathTableInfoEntry
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "
                    Path Table in the form of an array table containing
                    the path information foreach RRP device on the network
                    "
    ::= { rrp 30 }

PathTableEntry    OBJECT-TYPE
    SYNTAX          PathTableInfoEntry
    MAX-ACCESS      read-only
    STATUS          mandatory
    DESCRIPTION     "Row of Path Table Information"
    INDEX           { PathTableIndex }
    ::= { PathTableInfo 1 }

PathTableInfoEntry ::=  SEQUENCE {
                    pathDevAddress                      INTEGER,
                    pathHopCntRp1                       INTEGER,
                    pathHopCntRp2                       INTEGER,
                    pathPreferRp                        INTEGER,
                    pathDstRp                           INTEGER,
                    pathDevState                        INTEGER,
                    pathMACAddress                      MacAddress,
                    pathRp1Info                         INTEGER,
                    pathRp2Info                         INTEGER,
                    pathVersion                         INTEGER,
                    pathDevType                         INTEGER,
                    pathDevDesc                         VISIBLE STRING (SIZE(1..16)),
                    pathDevUID                          Counter64,
                    pathDevUIDRp1                       Counter64,
                    pathDevUIDRp2                       Counter64
                    }

END
-- *********************************************************************
--   EOF
-- *********************************************************************
```

Bibliography

IEC 61158 (all parts), *Industrial communication networks – Fieldbus specifications*

IEC 61158-4-21: 2010, *Industrial communication networks – Fieldbus specifications – Part 4-21: Data-link layer protocol specification – Type 21 elements*

ISO/IEC 9314-3:1990, *Information processing systems – Fibre distributed Data Interface (FDDI) – Part 3: Physical Layer Medium Dependent (PMD)*

IEEE 802.1D:2004, *IEEE Standard for Local and Metropolitan Area Networks – Media access control (MAC) Bridges*

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

ANSI X3.263:1995, *Fibre Distributed Data Interface (FDDI) – Token Ring Twisted Pair Physical Layer Medium Dependent (TP-PMD)*

This page deliberately left blank

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™