INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.9961

Amendment 2

(04/2014)

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Access networks – In premises networks

Unified high-speed wire-line based home networking transceivers - Data link layer specification

Amendment 2

CAUTION !

PREPUBLISHED RECOMMENDATION

This prepublication is an unedited version of a recently approved Recommendation. It will be replaced by the published version after editing. Therefore, there will be differences between this prepublication and the published version.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU [had/had not] received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at http://www.itu.int/ITU-T/ipr/.

Amendment 2 to Recommendation ITU-T G.9961 (2010)

Unified high-speed wire-line based home networking transceivers - Data link layer specification: Amendment 2

Summary

Amendment 2 to Recommendation ITU-T G.9961 (2010) contains the following:

1) Revision of text for clause 8.3.3.4.3 “CBTS back-off rules”.
2) Addition of working text for new clause 8.3.8 “Extended acknowledgements”.
3) Revision of the text for clause 8.5.3 “Routing of ADPs”.
4) Addition of working text for new clause 8.6.2.4 “Bandwidth update protocol for prioritized connections” and revision of clause 8.6.2.4.1.1 “Format of BU_BWUpdate.req”.
5) Revision of the text for clause 8.8.4 “TXOP descriptor”.
6) Revision of text in Table 8-88 in clause 8.10.1.1.
7) Revision of the text for clause 8.12.1.2 “Establishment of a data connection”.
8) Revision of the text for clause 8.17 “DLL multicast stream”.
9) Addition of text for new clause 8.20 “Metrics acquisition”.
10) Addition of working text for new Annex X “Test vectors”.

Amendment 2 to Recommendation ITU-T G.9961 (2010)

Unified high-speed wire-line based home networking transceivers - Data link layer specification: Amendment 2

1 Revise clause 8.3.3.4.3 "CBTS back-off rules" as follows:

8.3.3.4.3 CBTS back-off rules

All nodes contending in a CBTS shall use the back-off rules described in this clause in the CW. In the general case, CW immediately follows the PRS, as shown in Figure 8-16. The size of CW is expressed in the number of ITS. The valid values for the maximum range of the CW are defined in Table 8-7, the value of ITS is defined in clause 8.4. If PR signals are not required, the CW shall start right after the INUSE signal slot, as described in clause 8.3.3.4.6, or at the beginning of the CBTS, if INUSE is not used.

Each node shall maintain the following back-off parameters for each MA priority of the frame that node intends to transmit:

- back-off-counter (BC);
- defer counter (DC); and
- back-off stage counter (BSC).

The BC determines the number of ITS the node has to wait before it begins the transmission. The DC keeps track of the number of consecutive times a node can lose contention before changing the back-off parameters. The BSC keeps track of the back-off stage to enable the selection of BC and DC when the back-off stage changes.

Nodes that are allowed to compete in the CW shall use their back-off parameters for that MA priority, and act according to the following rules before starting a transmission in a CBTS:

1) If the BC is zero, the node shall start transmitting its frame within a time window of TX_ON microseconds after the start of the first ITS of the CW.
2) If the BC is not zero, the node shall decrement its BC upon completion of each ITS in which it detects no transmission.
3) If, upon completion of certain ITS, the value of BC is zero, the node shall start transmitting its frame within a time window of TX_ON microseconds after the end of the ITS.
4) If a node detects a transmission during an ITS, it shall not transmit in this CBTS and shall do the following:
   - The node shall decrement the DC.
   - If the DC is zero and BSC is less than BSC_{max}, the node shall increment the BSC. If the DC is zero and BSC is equal to BSC_{max}, the node shall maintain the current BSC. It shall then set DC to $DC_{max}(BSC)$ and BC to a random value in the range of $(0, NCW_{max}(BSC) – 1)$.
   - If the DC is greater than zero, the node shall decrement the BC.

Nodes that have inferred a collision (see clause 8.3.3.4.9) shall increment the BSC if BSC is less than BSC_{max}. It then sets DC to $DC_{max}(BSC)$ and BC to a random value in the range of $(0, NCW_{max}(BSC) – 1)$.

After initialization and upon successful transmission, nodes shall initialize BSC to 1, DC to $DC_{max}(1)$ and BC to a random value in the range (0, $NCW_{max}(1) - 1$).

As a default behaviour, after initialization and upon successful transmission, nodes shall initialize BSC to 1, DC to $DC_{max}(1)$ and BC to a random value in the range (0, $NCW_{max}(1) - 1$). This behaviour of the BSC, can be overridden by indicating so by the "Gradual BSC decrease indication flag" flag, in Table 8-85.3. If this flag is set to one, then after initialization and upon successful transmission, nodes shall decrement BSC by 1, unless BSC is already 1, in which case BSC shall be maintained as 1. In this situation, the recommended values for the values of $DC_{max}(BSC)$ are shown in Table 8-6.1, i.e., DC parameters are invalid for back-off rules.

Table 8-6.1 – $DC_{max}(BSC)$ values

BSC	$DC_{max}(BSC)$	$NCW_{max}(BSC)$
1	1	8
2	1	16
3	1	32
4	1	64

Table 8-7 shows the ~~valid~~ default values of $DC_{max}(BSC)$ and $NCW_{max}(BSC)$. These ~~valid~~ default values are used for all MA priorities. ~~BSC_{max} shall be 4.~~

The default values in Table 8-7 can be overridden by using the contention window (CW) information sub-field of the auxiliary information field in the MAP, as described in clause 8.8.5.11.

Table 8-7/~~G.9961~~ – ~~Valid~~ Default $DC_{max}(BSC)$ and $NCW_{max}(BSC)$ values

BSC	$DC_{max}(BSC)$	$NCW_{max}(BSC)$
1	1	8
2	2	16
3	4	32
4	16	64
~~NOTE – other values of BSC, DC_{max} and NCW_{max} are for further study.~~		

If a node that is allowed to contend in a CBTS has an MPDU ready to transmit after the start of the CW, it is still allowed to contend with this MPDU using the back-off procedure defined in this clause only if the MPDU's MA priority is equal to or higher than the MA priority that won the priority resolution. The node shall pick the BC random value for the ITS in the CW in the same way as nodes that had the frame ready to transmit prior to the start of the CW, and shall start decrementing the BC from the ITS where the frame was ready for transmission. The BC, DC, BSC values that shall be used are of the frame's MA priority value.

2 Add new clause 8.3.8 "Extended acknowledgements" as follows:

8.3.8 Extended Acknowledgements

Extended acknowledgements between two nodes may be used to improve throughput and reduce unnecessary retransmissions due to the lack of sufficient number of bits in the regular acknowledgement frame.

The BMSG frame is used by a source node to indicate to the destination node that it supports reception of an extended acknowledgement. The destination node that supports an extended acknowledgement can choose one of the following options on a per frame basis depending on its specific requirement:

1. It can send a regular acknowledgement.

2. It can send an extended acknowledgement.

3. In case of bidirectional transmissions, it can send a BACK frame, if it has additional data to transmit and doesn't need to send an extended acknowledgement.

A source node can offer a destination node this choice by sending a BMSG frame with the -BTXEF set to one, EXTACKGR bit set to one and BTXGL set to a non-zero value in any of the BMSG frames. In this case, as BTXGL ≠ 0, the destination node may send a BACK frame, a regular ACK frame or an extended ACK frame that fits within the granted time duration by the source node. The BTXGL shall at least include time for the destination node to send an extended ACK. Specifically, the destination node's response to the BMSG shall be as shown in Table 8-13.1:

Table 8-13.1 – Extended acknowledgement settings

BTXEF	BTXGL	Frame transmitted by the destination node
1	2 symbols for destination node	ACK or EACK
1	>2 symbols for destination node	ACK or EACK or BACK (Note 1)
NOTE 1 – This row applies to bidirectional transmissions.		

The BMSG PHY frames shall use the format described in Tables 7-47 and 7-53 of [ITU-T G.9960], and the BACK PHY frames shall use the format described in Tables 7-48 and 7-54 of [ITU-T G.9960], in which the PHY frame header contains $2 \times PHY_H$ information bits (EHI bit, in the PHY frame header, is set to one, see clause 7.1.2.3.1.7 of [ITU-T G.9960]).

The extended ACK frames shall use the format described in Table 7-54.1 of [ITU-T G.9960], in which the PHY frame header contains $2 \times PHY_H$ information bits (EHI bit, in the PHY frame header, is set to one, see clause 7.1.2.3.1.7 of [ITU-T G.9960]).

An exchange of BMSG and BACK/ACK/extended ACK frames forms a bidirectional frame sequence that shall last strictly inside the boundaries of the particular TXOP or TS assigned in the MAP for the node sourcing the bidirectional transmission. When using an extended acknowledgement, only immediate acknowledgement is allowed (the valid values of RPRQ field are 00 and 01 only).

An extended acknowledgement may be initiated by either a source node or a destination node using one of the following methods:

- A destination node transmits to the source node, in response to a MSG frame requesting immediate acknowledgement, an ACK frame with the EXTACKRQ bit set to one.
- A source node transmits to the destination node a BMSG frame with the BTXGL field set to a non-zero value, BTXEF bit set to one and EXTACKGR bit set to one.

If a source node requested by a destination node to initiate extended acknowledgements accepts the request, it shall indicate that the request is granted and shall initiate bidirectional transmission by transmitting a BMSG frame that initiates extended acknowledgements. Alternatively, the source node requested to initiate extended acknowledgement may decline the request. In this case it indicates that the extended acknowledgement request is declined by continuing to send BMSG frames with the EXTACKGR set to zero, instead of the BMSG frame that initiates extended acknowledgements.

A source node may initiate extended acknowledgements autonomously, without a request from the destination node. A source node may terminate extended acknowledgements at any time and re-start them again. The destination node may indicate to the source node when the extended acknowledgements may be stopped by setting the EXTACKRQ to zero, while the decision is up to the source node.

A destination node responds to the BMSG frame that initiates extended acknowledgements by one of the following ways:

- transmitting a BACK frame that contains data in the payload intended for the source node. The BACK frame additionally contains acknowledgement information for data previously transmitted by the source node. In the BTXRL field of the frame header the destination node indicates the requested duration of the next BACK frame it expects to transmit.
- transmitting an ACK frame
- transmitting an extended ACK frame

The destination node may indicate that extended acknowledgement is not needed any further (advice for termination of extended acknowledgement) by setting the BTXRL = 0 in the BACK frame or EXTACKRQ = 0 in the ACK frame. In response, the source node may terminate extended acknowledgement.

The maximum duration of a BACK frame is determined by the source node in the BTXGL field of the PHY-frame header. The destination node only indicates the desired duration of BACK frame in the BTXRL field of the PHY-frame header of the previous BACK frame, but the final decision on the BACK frame duration limit (including the following IFG) is done by the source node.

A responding BACK frame shall be transmitted $T_{BM2BAIFG}$ after the BMSG frame. The Imm-ACK frame shall be transmitted T_{AIFG} after the BMSG frame. In all of the following frame sequences:

- BMSG followed by a BACK
- BMSG followed by an Imm-ACK

if the transmitter of the first frame has no knowledge of the 'receiver specific' AIFG (see clause 8.6.1.1.4.1 and clause 8.6.4.3.1) or if the first frame in any of the above frame sequences includes less than MIN_SYM_VAR_AIFG symbols, the gap between this frame and the following frame shall be T_{AIFG-D} (see clause 8.4), otherwise the gap shall be T_{AIFG}. The parameter MIN_SYM_VAR_AIFG is defined in clause 8.4, for each media. The transmitter indicates usage of either T_{AIFG} or T_{AIFG-D} by using the AIFG_IND bit in the PHY-frame header (see clause 7.1.2.3.2.2.16 of [ITU-T G.9960]).

Extended acknowledgement can be used in CFTXOP, STXOP, and CBTXOP. The source node shall ensure that the total duration of the bidirectional frame sequence does not violate the boundaries of the TXOP or the maximum allowed duration of the TS. Particularly:

– if extended acknowledgement is used in a CFTXOP, the last frame in the sequence shall end at least T_{IFG_MIN} before the end of the CFTXOP;
– if extended acknowledgement is used in a CFTS or in a CBTS, the last frame in the sequence shall end at least T_{IFG_MIN} before the end of the Max_TS_Length assigned in the MAP for the TS and at least T_{IFG_MIN} before the end of the TXOP where this TS is defined.

When the bidirectional transmission is terminated with a BMSG frame with BTXEF = 1 and BTXGL ≠ 0, the total duration of the frame sequence shall include this BTXGL value, regardless of the actual duration of the last BACK frame, acknowledgement frame or extended acknowledgement frame.

Nodes detecting a bidirectional transmission shall stay silent until the end of the bidirectional transmission sequence or until the expiration of the Max_TS_Length of the corresponding TS, whichever comes first.

Extended acknowledgement is not allowed when RTS/CTS is used.

3 Revise the text of clause 8.5.3 "Routing of ADPs" (from G.9961 corr2) as follows:

8.5.3 Routing of ADPs

Each node shall inform the domain master about the nodes of its domain it has detected as defined in clause 8.6.4.3.

Each node can have one or more applications associated with its AE (above its A-interface). Each application is identified by a unique 6-octet MAC address. Each node shall maintain the full list of MAC addresses associated with applications above its A-interface as well as its own MAC address. This list is referred to as the local address association table (LAAT).

NOTE – The list of MAC addresses associated with applications above its A-interface for a node can be populated by learning, or directly programmed by the DLL management entity.

Each node shall also maintain the list of MAC addresses associated with the AEs of other nodes in the domain and the MAC addresses of those nodes. This list is referred to as a remote address association table (RAAT). Each node provides its local AAT to the domain master and other nodes of the domain using topology management messages as described in clause 8.6.4.3.

The address association table (AAT) is formed by the aggregation of LAAT and RAAT.

Whenever a node receives an ADP from the A-interface, it uses its AAT to determine if the ADP is intended for the node itself (local in-band management message, see Annex A) or for an AE associated with another node.

• If the ADP is intended for a remote AE or is an in-band management message addressed to a different node (case B of Table 8-14.1), the node shall determine the destination DEVICE_ID of the node in its domain through which the remote AE can be reached and send the corresponding ADP directly or via relay nodes to this node. This destination DEVICE_ID is provided to the Flow Mapper (see Figure 8-2) and is further reached either directly or via relays.

- If the ADP is intended for a group MAC address belonging to the AEs of different nodes of the domain (case D of Table 8-14.1), the node shall associate this ADP with a destination MSID and it shall send the APDU using DLL multicast transmission. The node may send the APDU to the appropriate nodes using unicast transmissions until the DLL multicast paths toward the appropriate nodes are established. The node may send the APDU using a combination of DLL multicast and DLL unicast transmissions until the relevant DLL multicast path is established.

  NOTE 1 – The association between the group of MAC addresses and addressed nodes is provided by the DLL management entity. The mechanism of this association is vendor discretionary and may be based on various multicast protocols, such as IGMP.

- If the destination address of the ADP is a standard broadcast address ($FFFFFFFFFFFF_{16}$) (case E of Table 8-14.1), then the BRCTI bit in the LFH of the LLC frame carrying the corresponding APDU shall be set to one, so that the APDU will be broadcast to all nodes in the domain using the procedure described in clause 8.5.4. If the EtherType of the ADP equals $22E3_{16}$, the corresponding APDU shall also be forwarded to the local DLL management entity.

  NOTE 2 – For ADP with EtherType different from $22E3_{16}$ and the standard broadcast address as the DA of that ADP, sending the corresponding APDU to the local DLL management entity is vendor discretionary.

- If the destination address of a received ADP is found in the local AAT and it is not the MAC address of the node (case A of Table 8-14.1), the ADP shall be dropped without notification.
- If the destination address of a received ADP is the MAC address of the node (case C of Table 8-14.1), the node shall pass the corresponding APDU to its DLL management entity.
- If the destination address of a received ADP is the reserved MAC address 01-19-A7-52-76-96 (case F of Table 8-14.1), the node shall pass the corresponding APDU to its DLL management entity
- If the destination MAC address corresponds to a unicast MAC address and the destination node cannot be inferred from previous rules (not covered in cases A, B, C and F), then the BRCTI bit in the LFH of the LLC frame carrying the corresponding APDU shall be set to one, so that the APDU will be broadcast to all nodes in the domain using the procedure described in clause 8.5.4. (case G of Table 8-14.1)
- If the destination MAC address corresponds to a group MAC address for which the destination nodes cannot be inferred or a group MAC address intended to reach all the nodes of the domain (case H of Table 8-14.1), then the BRCTI bit in the LFH of the LLC frame carrying the corresponding APDU shall be set to one, so that the APDU will be broadcast to all nodes in the domain using the procedure described in clause 8.5.4.

Table 8-14.1 – Routing of ADPs

Case	Ethernet frame type	ADP Destination address	Routing	Example
A	Unicast frame	In LAAT, except node's MAC address	Drop the message	Any kind of traffic
B	Unicast frame	In RAAT	Look for the DestinationNode defined for this DA	Normal routing of frames coming through the A interface (can be normal Ethernet or remote in-band messages)
C	Unicast frame	Node's MAC address	Send to DLL management	Local in-band message
D	Multicast frame	Multicast address mapped to known destination device(s)	The node has the choice to treat this multicast transmission as several DLL unicast transmissions or using a DLL multicast stream	IGMP/MLD Ethernet frames
E	Broadcast frame	Broadcast address	If EtherType = $22E3_{16}$ send to DLL management treat this broadcast transmission using BRT (BRCTI=1; DestinationNode = BROADCAST_ID) and route following the BRT rules	Normal broadcast
F	Unicast frame	Reserved address	Send to DLL management	
G	Unicast Frame	Destination MAC address not covered by cases A, B, C and F	Treat this case as a broadcast transmission using BRT (BRCTI=1; DestinationNode = BROADCAST_ID) and route following the BRT rules	Any kind of traffic
H	Multicast Frame	• Destination device(s) cannot be inferred from the DA or • Frame intended for all devices	Treat this case as a broadcast transmission using BRT (BRCTI=1; DestinationNode = BROADCAST_ID) and route following the BRT rules	Multicast protocol (IGMP/MLD) control frames

4 Add new clause 8.6.2.4 "Bandwidth update protocol for prioritized connections" as follows:

8.6.2.4 Bandwidth update protocol for prioritized connections

This clause defines the mechanism used by the nodes to inform the domain master on the actual status of its connection queues. This protocol is supported by management messages described in clause 8.6.2.4.1.

A node should follow the bandwidth update protocol when a node does not have direct visibility with the domain master (the BRURQ field in the PHY-frame header of transmitted PHY frames is not received by the domain master).

In addition, any node should use this protocol when

- The domain master does not allocate in the MAP enough TXOPs allowing to transmit user priorities queued in the node
- The node wants to inform the domain master on the status of a particular connection

The domain master may take this information into account when assigning resources for a given connection. As prioritized connections are not QoS guaranteed, the domain master may change bandwidth allocations on its own discretion.

To inform the DM about the necessity to allocate bandwidth for prioritized connections a node shall send a BU_BWUpdate.req message to its domain master including the information about the reported connection status (user priority in the connection queue, bandwidth request update).

Upon reception of BU_BWUpdate.req, the domain master shall send to the reporting node a BU_BWUpdate.cnf message acknowledging the reception of the information. If a node doesn't receive the BU_BWUpdate.cnf message within 200 ms, it may repeat the report.

If the request is acknowledged by the DM, the node shall refrain from reporting a new bandwidth update for at least the next 1 second.

> **NOTE** – A node should report the status of its queues when traffic conditions change but taking care not to flood the DM with report messages. If a node has enough resources with the current allocation by the domain master, it should only send a report when its traffic requirements change.

8.6.2.4.1 Bandwidth update protocol messages

The following sub-clauses specify the messages that are needed to support the bandwidth update protocol.

8.6.2.4.1.1 Format of BU_BWUpdate.req

This message is sent by the reporting node to the Domain Master and contains the user priorities, the bandwidth request update for the given connection by means of BRURQ indication, and PHY data rate used by the transmitter.

The format of the MMPL of the BU_BWUpdate.req shall be as shown in Table 8-44.1.

Table 8-44.1 – Format of the MMPL of the BU_BWUpdate.req message

Field	Octet	Bits	Description
DeviceID	0	[7:0]	DEVICE_ID of the originating node.
PRIORITY_QUEUE	1	[7:0]	Priority queue corresponding to the BRURQ we are reporting
Priorities	2	[7:0]	User priority bit mask. Each bit represents one user priority. Bit 0 represents user priority 0 and bit 7 user priority 7. Each bit signals the presence of at least one APDU with this user priority in the selected connection
BRURQ	4&5	[15:0]	See in clause 7.1.2.3.2.2.19 (NOTE)
TxRate	Variable	See Table 8-33	The actual PHY data rate used by the transmitter, specified in bits per second for each channel estimation window, based on the bit loading per symbol, the symbol time, the FEC rate and the number of repetitions. The format of the TX rate field is described in Table 8-33. Note that the TX Rate should be specified per each channel estimation window.
NOTE – The reporting node may set this field to zero in order to indicate that the DM may release the resources allocated for the connection that is being reported.			

8.6.2.4.1.2 Format of BU_BWUpdate.cnf

This message is sent by the Domain Master to the reporting node after it has assessed whether the bandwidth allocation for the node can be provided.

The format of the MMPL of the BU_BWUpdate.cnf shall be as shown in Table 8-44.2.

Table 8-44.2 – Format of the MMPL of the BU_BWUpdate.cnf message

Field	Octet	Bits	Description
DeviceID	0	[7:0]	DEVICE_ID of the reporting node.
PRIORITY_QUEUE	1	[7:0]	Priority queue specified by the reporting node in the BU_BWUpdate.req message
StatusCode	3	[7:0]	Status of the request for bandwidth allocation: • 00_{16} = Success. • 01_{16} = Failure – Insufficient resources. • 02_{16} = Failure – Insufficient resources for the requested priority • 03_{16} – FF_{16} = Reserved.

5 Revise the text of clause 8.8.4 "TXOP descriptor" as follows:

8.8.4 TXOP descriptor

Each TXOP is described by at least one TXOP descriptor. A TXOP descriptor is composed of a basic TXOP descriptor that may be extended by one or more additional TXOP descriptor extensions (see clauses 8.8.4.1.1 to 8.8.4.1.3). TXOP descriptor extensions supply additional information like scheduling information, timing information and TXOP attributes.

Basic TXOP descriptors and TXOP descriptor extensions are each four octets in length.

A TXOP descriptor represents the right of a certain node or a set of nodes to transmit within a certain TXOP. A CFTXOP shall be described using a single TXOP descriptor. A CBTXOP shall be described by either a single TXOP descriptor (see clause 8.3.3.4.5.3) or by multiple TXOP descriptors (see clause 8.3.3.4.5.2). A STXOP shall be described using several TXOP descriptors representing the TSs within the STXOP.

The domain master shall not assign more than 127 TXOP descriptors in the MAP, describing TSs, within a single STXOP (including CBTXOP).

The differentiation between different TXOPs shall be done via the TXOP attributes extension, which shall be appended to the last TXOP descriptor of a TXOP. The TXOP attributes extension supplies the timing information for the TXOP (see clause 8.8.4.1.1). A node associated with a TXOP or a TS is uniquely identified in a TXOP descriptor by the *SID* field, which shall be set to the DEVICE_ID of the node as was assigned by the domain master.

A flow associated with a TXOP or a TS is uniquely identified in a TXOP descriptor by the combination (SID, FLOW_ID). A FLOW_ID is a unique identifier of a flow associated with the SID.

A user priority associated with a TXOP or TS is uniquely identified in a TXOP descriptor by the tuple (SID, PRI). The PRI value shall represent the lowest MPDU priority that may be sent in the TXOP or TS.

A special TXOP is specified for assignment to nodes for transmitting traffic associated with bandwidth managed DLL multicast stream. These TXOPs are assigned to a DLL multicast stream by using the 'Multicast Indication' field in the basic TXOP descriptor, combined with 'FLOW_ID/PRI/MSID' field in the basic TXOP descriptor to identify the MSID of the stream and the 'SID' field in the basic TXOP descriptor to identify an intermediate node in the DLL multicast stream that is responsible for the transmission of data associated with the DLL multicast stream. The 'DID/Originating Node' field in the TXOP descriptor shall comtain the originating node of the DLL multicast stream and together with the MSID field, uniquely identifies the DLL multicast stream in the domain.

Table 8-63 describes the basic TXOP descriptor. When the extension bit is set, the TXOP descriptor shall have an extension, as described in clause 8.8.4.1. Different types of TXOP descriptor extensions are distinguished by extension type.

Table 8-63 – Basic TXOP descriptor format

Field	Octet	Bits	Description
SID	0	[7:0]	SID = 1-250 identifies the DEVICE_ID of the node assigned to the TXOP. SID = 0, 255 indicates special values for the TXOP descriptor (see clause 8.8.4.2).
DID/Originating Node	1	[7:0]	If Multicast Indication = 0: • DID = 0 indicates that the DID of the destination node of the flow is not known to the domain master. • DID > 0 indicates the destination node for the flow. DID shall be set to the DEVICE_ID as described in Table 8-61. If Multicast Indication = 1: This field identifies the originating node of the DLL multicast stream for which this TXOP is assigned.
Multicast Indication/MAP type	2 and 3	[0]	If this field is a special TXOP descriptor of a MAP (see clause 8.8.4.2) it indicates the type of MAP that shall be transmitted: 0 indicates MAP-A, 1 indicates MAP-D. If this field is not a special TXOP descriptor of a MAP this field contains the multicast indication: 1 indicates multicast/broadcast DID, 0 otherwise.
PR signal required		[1]	This bit instructs nodes contending for transmission in a CBTS whether to use the PR signal: 0 – PR signal shall not be used. 1 – PR signal is required.
CBTS Closure Mode		[3:2]	This field instructs nodes where to close a CBTS that was used for transmission (see in clause 8.3.3.4.5): 00 – Duration-based. 01 – Timeout-based from frame sequence start. 10 – Timeout-based from CBTS start. 11 – Reserved by ITU-T.
~~Reduced MAP~~		~~[4]~~	~~0 – TXOP for a complete MAP.~~ ~~1 – TXOP for a reduced MAP. See clause 8.8.1.~~
Reserved		[5:4]	Reserved by ITU-T (Note).
FLOW_ID/PRI/MSID		[13:6]	If Multicast Indication = 0: • Identifies the flow or the user priority associated with the TXOP/TS. • Valid values for user priority assignments are 0-7 —Valid values for FLOW_ID assignments are 8-2504 If Multicast Indication = 1: • Valid values for MSID assignments are 1-250 • Value 0 is reserved by ITU-T Values 251- 254 are reserved by ITU-T Value 255 indicates special values for the TXOP descriptor (see

Table 8-63 – Basic TXOP descriptor format

Field	Octet	Bits	Description
			clause 8.8.4.2).
Last_in_Group		[14]	1 indicates the last TS of a group of TSs in STXOP, 0 otherwise. Shall be set to zero for CFTXOP.
Extension		[15]	0 – No extension is present. 1 – This TXOP descriptor contains an extension.
NOTE – Bits that are reserved by ITU-T shall be set to zero by the transmitter and ignored by the receiver.			

Several TSs within the same STXOP can be grouped together to specify common attributes for these TSs via a group information extension (see clause 8.8.4.1.3). Grouping of several TSs shall be done by setting the Last_in_Group indication in the TXOP descriptor of the last TS of the group. Groups are implicitly numbered according to their appearance in the MAP. The first group shall be identified as group number one and so on. If a group contains only one TS, the descriptor of this TS shall have its Last_in_Group bit set to one.

6 Revise Table 8-88 "OPCODEs of management messages" in clause 8.10.1.1 as follows:

Table 8-88 – OPCODEs of management messages

Category	Message name	OPCODE (hex)	Description	MMPL Reference
Admission (01X)	ADM_NodeRegistrRequest.req	010	Registration request	Clause 8.6.1.1.4.1
	ADM_DmRegistrResponse.cnf	011	Registration response	Clause 8.6.1.1.4.2
	ADM_NodeResignRequest.req	012	Resignation request	Clause 8.6.1.1.4.3
	ADM_DmResign.cnf	013	Registration announcement	Clause 8.6.1.1.4.4
	ADM_DmForcedResign.req	014	Forced resignation request	Clause 8.6.1.1.4.5
	ADM_NodeReRegistrRequest.req	015	Periodic re-registration request	Clause 8.6.1.1.4.6
	ADM_DmReRegistrResponse.cnf	016	Periodic re-registration response	Clause 8.6.1.1.4.7
	ADM_DmReRegistrInitiate.ind	017	Re-registration initiation request	Clause 8.6.1.1.4.8
	ADM_NodeReportMAPD.ind	018	Report the reception of a MAP-D with matching domain name	Clause 8.6.6.1.4.1
	ADM_NodeReportMAPA.ind	019	Report the reception of a MAP-A with matching DNI	Clause 8.6.6.1.4.2

AKM (02X)	AUT_NodeRequest.req	020	Request for authentication	Clause 9.2.5.1.1
	AUT_Promp.ind	021	Delivers authentication prompt	Clause 9.2.5.1.2
	AUT_Verification.res	022	Authentication prompt verification	Clause 9.2.5.1.3
	AUT_Confirmation.cnf	023	Authentication confirmation message	Clause 9.2.5.1.4
	AKM_KeyRequest.req	024	Request for secure communication with another node(s)	Clause 9.2.5.2.1
	AKM_NewKey.req	025	Message delivers the encryption key to the Supplicant node	Clause 9.2.5.2.2
	AKM_KeyConfirmation.req	026	Message delivers the encryption key to the Addressee node(s)	Clause 9.2.5.2.3
	AKM_KeyUpdate.req	027	Request for re-authentication and update the keys	Clause 9.2.5.3.1
	AKM_KeyAck.cnf	028	Addressee confirmation that encryption key was delivered	Clause 9.2.5.2.3
	SC_DMRes.req	029	Request to resign a node from the domain	Clause 9.2.5.2.5
	SC_DMRes.cnf	02A	Confirmation of resignation from the domain master	Clause 9.2.5.2.6
	AKM_KeyAddRequest.req	02B	Request to join a node to a multicast group	Clause 9.2.5.2.1.1
	AKM_DomainKeyUpdate.ind	02C	Indication to update the domain-wide encryption keys	Clause 9.2.5.3.2
	AKM_NewKey.ind	02D	Indication that the new encryption key is available for use	Clause 9.2.5.2.7
Topology maintenance (03X)	TM_NodeTopologyChange.ind	030	Topology report from a node	Clause 8.6.4.3.12.1
	TM_NodeTopologyChange.req	031	Request sent by the domain master to a particular node requesting its topology report	Clause 8.6.4.3.2

	TM_NodeTopologyChange.cnf	032	Topology report from a node in response to the message TM_NodeTopologyChange.req	Clause 8.6.4.3.3
	TM_DomainRoutingChange.ind	033~~1~~	Optimal routing update from the domain master	Clause 8.6.4.3.5
	TM_ReturnDomainRouting.req	034~~2~~	Request for routing update from the node to the domain master	Clause 8.6.4.3.6
	TM_ReturnDomainRouting.cnf	035~~3~~	Reply on routing request by the Domain master	Clause 8.6.4.3.7
	TM_DMBackup.ind	036~~4~~	Topology report from a node sent by backup domain master to a node	Clause 8.6.4.3.4
Power-line coexistence with alien networks (04X)	Reserved for use by G.9972 [2]			
Multicast Binding (05X)	MC_GrpInfoUpdate.ind	050	Multicast Binding Information update	Clause 8.16.5.1
	MC_GrpInfoUpdate.cnf	051	Multicast binding information update confirmation	Clause 8.16.5.2
	MC_GrpRemove.req	052	Multicast leave request from the transmitter	Clause 8.16.5.3
	MC_GrpRemove.cnf	053	Multicast leave confirmation from the receiver	Clause 8.16.5.4
	DMC_Path.req	054	DLL multicast path establishment request	Clause 8.17.6.1
	DMC_Path.cnf	055	DLL multicast path establishment confirmation	Clause 8.17.6.2
	DMC_PathReject.cnf	056	DLL multicast path establishment rejection	Clause 8.17.6.3
	DMC_EnforcePath.req	057	DLL multicast enforced path establishment request	Clause 8.17.6.4
	DMC_ReleasePath.req	058	A request to release a DLL multicast client node from its MSID	Clause 8.17.6.5

	DMC_ReleasePath.cnf	059	Confirmation of the release of a DLL multicast client node from its MSID	Clause 8.17.6.6
	DMC_PathAlive.ind	05A	DLL multicast path alive indication	Clause 8.17.6.7
	DMC_BrokenLink.ind	05B	DLL multicast broken link indication	Clause 8.17.6.8
Domain Master Selection and Backup Domain Master (06X)	DM_Handover.req	060	Domain master role handover request	Clause 8.6.6.5.1
	DM_Handover.cnf	061	Domain master role handover confirmation	Clause 8.6.6.5.2
	DM_Handover.ind	062	Domain state update	Clause 8.6.6.5.3
	DM_Handover.rsp	063	Domain state update confirmation	Clause 8.6.6.5.4
	DM_BackupAssign.req	064	Backup domain master assignment request	Clause 8.6.5.2
	DM_BackupAssign.cnf	065	Backup domain master assignment confirmation	Clause 8.6.5.2
	DM_BackupData.ind	066	Domain state update	Clause 8.6.5.2
	DM_BackupRelease.req	067	Release of a backup domain master	Clause 8.6.5.2
	DM_BackupRelease.cnf	068	Backup domain master release confirmation	Clause 8.6.5.2
Channel Estimation (07X)	CE_ProbeSlotAssign.reqRequest.ind	070	Channel estimation bandwidth assignment request	Clause 8.11.7.1
	CE_ProbeSlotRelease.reqind	071	Channel estimation bandwidth release request	Clause 8.11.7.2
	CE_ParamUpdate.reqind	072	Channel estimation parameters update request	Clause 8.11.7.1
	CE_ParamUpdateRequest.ind	073	Request for Cchannel estimation parameter requestupdate	Clause 8.11.7.4
	CE_PartialBatUpdate.reqind	074	Partial BAT update indicationrequest	Clause 8.11.7.5
	CE_ACESymbols.ind	075	Request for an ACE symbol attachment	Clause 8.11.7.6

	CE_ProbeSlotAssign.cnf	076	Channel estimation bandwidth assignment confirmation	Clause 8.11.7.7
	CE_ProbeSlotRelease.cnf	077	Channel estimation bandwidth release confirmation	Clause 8.11.7.8
	CE_ParamUpdate.cnf	078	Channel estimation parameters update confirmation	Clause 8.11.7.9
	CE_PartialBatUpdate.cnf	079	Partial BAT update confirmation	Clause 8.11.7.10
Neighbouring ~~N~~networks coordination (08X)	For ~~F~~further study	For ~~F~~further study	For ~~F~~further study	For ~~F~~further study
Inactivity scheduling (09X)	IAS_LongInactivity.req	090	Long inactivity scheduling request	Clause 8.3.6.1.1
	IAS_LongInactivity.cnf	091	Long inactivity scheduling confirmation	Clause 8.3.6.1.1
	IAS_ShortInactivity.req	092	Short inactivity scheduling request	Clause 8.3.6.2.1
	IAS_ShortInactivity.cnf	093	Short inactivity scheduling confirmation	Clause 8.3.6.2.1
Flow establishment (0AX)	~~CL_EstablishFlow.req~~Reserved	0A0	~~Flow establishment request~~ Reserved by ITU-T	~~§ 8.6.2.3.1~~
	~~CL_EstablishFlow.cnf~~Reserved	0A1	Reserved by ITU-T~~Flow establishment confirmation~~	~~§ 8.6.2.3.2~~
	FL_AdmitFlow.req	0A2	Flow admission request	Clause 8.6.2.3.8
	FL_AdmitFlow.cnf	0A3	Flow admission confirmation	Clause 8.6.2.3.9
	FL_AdmitFlow.ind	0A4	Flow admission indication	Clause 8.6.2.3.10
	FL_AdmitFlow.rsp	0A5	Flow admission acknowledgement	Clause 8.6.2.3.18
	FL_OriginateFlow.req	0A6~~4~~	Flow origination request	Clause 8.6.2.3.6
	FL_OriginateFlow.cnf	0A7~~5~~	Flow origination confirmation	Clause 8.6.2.3.7
Flow maintenance (0BX)	FL_ModifyFlowParameters.req	0B0	Modification of flow parameters and allocation	Clause 8.6.2.3.11
	FL_ModifyFlowParameters.cnf	0B1		Clause 8.6.2.3.12

	FL_ModifyFlowParameters.ind	0B2		Clause 8.6.2.3.15
	FL_ModifyFlowAllocations.req	0B3	Modification of flow allocation	Clause 8.6.2.3.17
	FL_ModifyFlowAllocations.cnf	0B4		Clause 8.6.2.3.18
Flow termination (0CX)	~~CL_TerminateFlow.req~~<u>Reserved</u>	0C0	~~Flow termination request and confirmation~~<u>Reserved by ITU-T</u>	~~§8.6.2.3.3~~
	<u>Reserved</u>~~CL_TerminateFlow.cnf~~	0C1	<u>Reserved by ITU-T</u>	~~§8.6.2.3.4~~
	<u>Reserved</u>~~CL_FlowTerminated.ind~~	0C2	<u>Reserved by ITU-T</u>	~~§8.6.2.3.5~~
	FL_TerminateFlow.req	0C3		Clause 8.6.2.3.13
	FL_TerminateFlow.cnf	0C4		Clause 8.6.2.3.14
	<u>FL_BrokenTunnel.ind</u>	<u>0C5</u>	<u>Indicate broken tunnel</u>	<u>Clause 8.6.2.3.19</u>
	<u>FL_BrokenTunnel.rsp</u>	<u>0C6</u>	<u>Response to indication</u>	<u>Clause 8.6.2.3.20</u>
	<u>FL_ReleaseTunnel.req</u>	<u>0C7</u>	<u>Request Release Tunnel</u>	<u>Clause 8.6.2.3.21</u>
	<u>FL_ReleaseTunnel.cnf</u>	<u>0C8</u>	<u>Confirm Release Tunnel</u>	<u>Clause 8.6.2.3.22</u>
	<u>FL_DM_RenewTunnel.req</u>	<u>0C9</u>	<u>DM renew tunnel request</u>	<u>Clause 8.6.2.3.23</u>
	<u>FL_DM_RenewTunnel.cnf</u>	<u>0CA</u>	<u>Confirm DM renew tunnel</u>	<u>Clause 8.6.2.3.24</u>
	<u>FL_RenewTunnel.req</u>	<u>0CB</u>	<u>Renew tunnel request</u>	<u>Clause 8.6.2.3.25</u>
	<u>FL_RenewTunnel.cnf</u>	<u>0CC</u>	<u>Confirm Renew tunnel</u>	<u>Clause 8.6.2.3.26</u>
	<u>FL_DeleteFlow.req</u>	<u>0CD</u>	<u>Delete Flow request</u>	<u>Clause 8.6.2.3.27</u>
	<u>FL_DeleteFlow.cnf</u>	<u>0CE</u>	<u>Confirm Delete Flow</u>	<u>Clause 8.6.2.3.28</u>
<u>Media Access Plan (0DX)</u>	<u>MAP</u>	<u>0D0</u>	<u>MAP message</u>	<u>Clause 8.8</u>
<u>Channel Estimation 2 (0EX)</u>	<u>CE_Request.ind</u>	<u>0E0</u>	<u>Channel estimation trigger</u>	<u>Clause 8.11.7.11</u>
	<u>CE_Initiation.req</u>	<u>0E1</u>	<u>Channel estimation initiation request</u>	<u>Clause 8.11.7.12</u>
	<u>CE_Initiation.cnf</u>	<u>0E2</u>	<u>Channel estimation initiation confirmation</u>	<u>Clause 8.11.7.13</u>
	<u>CE_ProbeRequest.ind</u>	<u>0E3</u>	<u>Request for PROBE frame transmission</u>	<u>Clause 8.11.7.14</u>

	CE_Cancellation.req	0E4	Channel estimation cancellation request	Clause 8.11.7.15
	CE_BatIdMaintain.ind	0E5	BAT ID maintenance	Clause 8.11.7.16
	CE_Cancellation.cnf	0E6	Channel estimation cancellation confirmation	Clause 8.11.7.17
	Reserved	0E7 – 0EF	Reserved by ITU-T	
Transmission Profile (0FX)	TP_TransmitPsdChange.req	0F0	Transmit PSD mask change request	Clause 8.6.9.1
	TP_TransmitPsdChange.cnf	0F1	Transmit PSD mask change confirmation	Clause 8.6.9.2
Neighbouring network coö~~o~~rdination (10X to 13~~1~~X)	NDIM_StartAlignmentProcedure.ind	100	Request to start a MAC cycle alignment procedure (DM to proxy node)	Clause 8.14.9.1
	NDIM_IDCCReserve.req	101	Slot reservation request	Clause 8.14.9.2
	NDIM_IDCCReserve.cnf	102	Slot reservation confirmation	Clause 8.14.9.3
	NDIM_ReportAlignment.req	103	Report on MAC cycle alignment	Clause 8.14.9.4
	NDIM_ReportAlignment.cnf	104	Confirm receiving NDIM_ReportAlignment.req	Clause 8.14.9.5
	NDIM_RemotePresence.req	105	Request to respond to ID_Presence~~.~~Request	Clause 8.14.9.6
	NDIM_RemotePresence.cnf	106	Permission to respond to ID_Presence~~.~~Request	Clause 8.14.9.7
	NDIM_Transmit.ind	107	DM to proxy node message to be transmitted to neighbouring domain	Clause 8.14.9.8
	NDIM_Receive.ind	108	Proxy node to DM message received from neighbouring domain	Clause 8.14.9.9
	NDIM_InterferenceReport.ind	109	Indication of interference detected	Clause 8.14.9.12
	NDIM_IDCC_Release.req	10A	Release Slot reservation	Clause 8.14.9.10
	NDIM_IDCC_Release.cnf	10B	Confirm receiving NDIM_IDCC_Release.req	Clause 8.14.9.11
	IDM_ClusterAlignment.req	120	DM informs other DMs about new cluster alignment	Clause 8.14.10.1

	IDM_ClusterAlignment.cnf	121	DM confirm receiving IDM_ClusterAlignment.req	Clause 8.14.10.2
	IDM_InterfNodesInfo.ind	122	Proxy node to neighbouring domains indication of interfering nodes	Clause 8.14.10.3
	IDM_CoördDomainsInfo.ind	123	Proxy node to neighbouring domains indication of coördinating nodes	Clause 8.14.10.5
	IDM_ShareUnallocSlot.req	124	Request to share unallocated slots	Clause 8.14.10.7
	IDM_ShareUnallocSlot.cnf	125	Confirmation of request to share unallocated slots	Clause 8.14.10.8
	IDM_ShareUnallocSlot.ind	126	Indication of status of the request to share unallocated slots	Clause 8.14.10.9
	IDM_RequestUnallocSlot.req	127	Request assignment of unallocated slots	Clause 8.14.10.10
	IDM_RequestUnallocSlot.cnf	128	Confirmation of request for assignment of unallocated slots	Clause 8.14.10.11
	IDM_RequestUnallocSlot.ind	129	Indication of status of the request for assignment of unallocated slots	Clause 8.14.10.12
	IDM_SwapAllocSlot.req	12A	Request to swap allocated slots	Clause 8.14.10.13
	IDM_SwapAllocSlot.cnf	12B	Confirmation of the request to swap allocated slots	Clause 8.14.10.14
	IDM_SwapAllocSlot.ind	12C	Indication of status of the request to swap allocated slots	Clause 8.14.10.15
	IDM_CoördPref.ind	12D	Indication of preferred coördination method	Clause 8.14.10.16
	IDM_DmChange.ind	12E	Indication to neighbouring domain masters that the DM of the domain sending the message has changed	Clause 8.14.10.17

	IDM_DniChange.ind	12F	Indication to neighbouring domain masters that the DNI of the domain sending the message has changed	Clause 8.14.10.18
	IDM_InterfNodesInfo.rsp	130	A message sent as a confirmation for a received IDM_ InterfNodesInfo.ind	Clause 8.14.10.4
	IDM_CoördDomainsInfo.rsp	131	A message sent as a confirmation for a received IDM_ CoördDomainsInfo.ind	Clause 8.14.10.6
Bandwidth update (14X)	BU_BWUpdate.req	140	Bandwidth update request	Clause 8.6.2.4.1.1
	BU_BWUpdate.cnf	141	Bandwidth update confirmation	Clause 8.6.2.4.1.2
Reserved	Reserved	150 – 7FF	Reserved by ITU-T	
MIMO (8XX – 9XX)	Reserved for use by G.9963 [x]	800 – 9FF		
Reserved	Reserved	A00-FFF	Reserved by ITU-T	

7 Revise clause 8.12.1.2 "Establishment of a data connection" as follows:

8.12.1.2 Establishment of a data connection

A data connection shall be established using the procotol described in clause 8.9.5.3, where the transmitter shall send a PHY frame with FT=MSG, CNN_MNGMT=0101, START_SSN=ACK_TX_WINDOW_START, no payload and RPRQ=01. The transmitter may advise the receiver about the required number of LPDUs that the receiver should buffer for this connection by setting the ADVISED_WIN_SIZE field.

If the receiver has resources to handle the new connection, it shall respond with a PHY frame with FT=ACK, RXRST_DATA=1. In this ACK frame, the receiver shall use the flow control fields FLCTRLT, FLCTRL and FLCTRL_CONN to provide additional flow control information, such as receiver buffer size or hold time. Once the protocol described in clause 8.9.5.3 is finished successfully, the transmitter may start sending PHY frames with data segments.

Following the protocol described in clause 8.9.5.3, if the receiver temporarily does not have resources to handle the new connection, it shall respond with a PHY frame with FT=ACK, RXRST_DATA=1, FLCTRLT=<Hold Time>, FLCTRL_CONN=0 and FLCTRL equal to the amount of time desired by the receiver.

If the receiver does not have resources to handle the new connection, it shall respond with a PHY frame with FT=ACK, RXRST_DATA=1, FLCTRLT=<Hold Time>, FLCTRL_CONN=0 and FLCTRL=31.

If the receiver has resources for the new connection, it shall respond with a PHY frame with FT=ACK, RXRST_DATA=1, FLCTRLT=<Status report>, FLCTRL_CONN=0~~1~~ and FLCTRL equal to the number of LPDUs that the receiver can buffer for this connection. The receiver may consider the value of the ADVISED_WIN_SIZE as proposed by the transmitter to set its FLCTRL field and RX_CONN_WIN_SIZE field. The transmitter shall set ACK_TX_CONF_WINDOW_SIZE (see clause 8.9.4.2) to the minimum of the value indicated in the RX_CONN_WIN_SIZE ~~FLCTRL~~ field and its own available window size (see clause 7.1.2.3.2.3.8 of [ITU-T G.9960]). The number of LPDUs that the receiver can buffer for this connection, indicated by the FLCTRL field during the lifetime of the connection, shall not exceed the maximum acknowledge window size that the receiver can support for the connection indicated by RX_CONN_WIN_SIZE during connection setup.

8 Revise the text of clause 8.17 "DLL multicast stream" (from G.9961 corr1) as follows:

8.17 DLL multicast stream

A source node that decides to establish a DLL multicast stream shall establish a multicast path toward each client of the DLL multicast stream. The paths toward the client nodes may include relay nodes that are bound to the path and the DLL Multicast Stream identification (MSID). The source node that establishes a DLL multicast group shall generate the DLL multicast stream identifier (MSID) that together with the DEVICE_ID of source of the DLL multicast stream uniquely identifies the DLL multicast stream. The members of a DLL multicast group are identified by the source node of the DLL multicast stream. The source of a DLL multicast stream, shall transmit the traffic of the DLL multicast stream to the members of the DLL multicast group according to established paths as described in the following ~~section~~clauses.

8.17.1 DLL multicast stream establishment

A G.hn node that determines that it has to transmit a multicast stream to client nodes in the domain, shall establish a path to each one of the client nodes. The source node that generates the DLL multicast stream shall first allocate an MSID that, together with the DEVICE_ID of the source node, shall uniquely identify the DLL multicast stream. Valid values of MSID are from 1 to 250. The source node shall also initialize the Transaction ID for that DLL multicast stream to zero. The source node shall increment the Transaction ID- for each new DLL multicast path it establishes for that DLL multicast stream.

The source node shall establish the path toward a client node as follows:

If the source node has a direct link to the client node according to the current unicast routing table, it shall send a DMC_Path.req message to the client node to bind it with the specified MSID multicast stream and the multicast stream MAC Address. The client node shall reply with a DMC_Path.cnf message that contains the same Transaction_ID that was specified in the DMC_Path.req message and shall bind itself to the established path identified by the MSID, the DEVICE_ID of the source node and the multicast MAC address (DA). The source node upon receiving the DMC_Path.cnf message shall bind the path and complete the path establishment procedure. If the source node does not receive a DMC_Path.cnf message after a vendor discretionary period, which is larger than MAX_WAIT_TIME, it may repeat the request through a new DMC_Path.req with a different Transaction_ID.

NOTE – DMC_Path.req should be sent using connections with acknowledgements in order to avoid long setup times for DLL multicast trees because of lost messages.

If the source node does not have a direct link to the client node, it shall determine the first relay node towards the client node according to the current unicast routing table and send a DMC_Path.req message to that node.

The DMC_Path.req message shall contain the following fields: the DEVICE_ID of the source node of the DLL multicast, the allocated MSID, the DEVICE_ID of the client node, the MAC address of the multicast stream, and the Transaction_ID, and the DEVICE_ID of the first relay node (Relay ID). If there is a direct link between the source node and the client node, the Relay ID field shall be set to the DEVICE_ID of the client node. The source node of the multicast stream shall address the DMC_Path.req message to the first relay node by setting the DA to the MAC address of that node.

A relay node that receives a DMC_Path.req message and has a direct link with the client node shall bind the DEVICE_ID of the source of the DLL multicast stream, the MSID, and the sender node's DEVICE_ID with the DEVICE_ID of the client endpoint node, and shall replace the DA of the LCDU of the DMC_Path.req message with the client node's MAC address and transmit the DMC_Path.req message to the client node.

A relay node that receives a DMC_Path.req message and doesn't have a direct link to the client node shall bind the DEVICE_ID of the source of the DLL multicast stream, the MSID, and the sender node's DEVICE_ID with the DEVICE_ID of the next relay node towards the client node according to the unicast routing table. It shall then replace the DMC_Path.req LCDU's DA by the MAC address of the next relay node and send the updated DMC_Path.req message to that node.

Upon reception of the DMC_Path.req message, the client node shall reply to the node that sent this message with a DMC_Path.cnf message and shall bind itself to the specified DLL multicast stream identified by the DEVICE_ID of the source DLL multicast stream, the MSID, and the sender node's DEVICE_ID.

A relay node that receives the DMC_Path.cnf message shall mark the binding of the DLL multicast stream path identified by the DEVICE ID of the source DLL multicast stream node and the MSID as valid. The relay node shall then append its DEVICE_ID to the Path_List field in the MMPL of the received DMC_Path.cnf message. The relay node shall transmit the updated DMC_Path.cnf message to the node from which it has received the DMC_Path.req message, which can be either a relay node or the node originating the DLL multicast stream.

Once the originating node receives the DMC_Path.cnf message, it has the complete path of this bound client from the received DMC_Path.cnf message. This completes the path establishment procedure. The source node may then start sending the multicast stream packets towards the client node(s) either directly or via the first relay node according to the established path.

Each relay node shall identify LLC frames corresponding to a DLL multicast stream according to the OriginatingNode and the MSID specified in the LFH. The relay node shall then relay any received LLC frames of that DLL multicast stream to all the nodes it has bound to this DLL multicast stream according to the binding information that it has configured during the DLL multicast stream path establishment. The relay node shall only relay LLC frames corresponding to a DLL multicast stream path for which its binding is marked as valid.

When the multicast source node or any other relay node in the DLL multicast paths receives an updated routing table, it shall not update the current multicast paths. A relay node shall correct an established multicast path only by explicit order received from the multicast source node as defined in clause 8.17.3.

8.17.2 DLL multicast stream establishment with bandwidth reservation

An originating node that decides to establish a DLL multicast stream shall also decide whether to have it 'BW reserved' or not. The 'BW reserved' attribute shall be carried in the DLL multicast stream protocol messages so all participating nodes will have that knowledge as well.

It is the responsibility of the originating node and the relay nodes in the stream to make sure new non-leaf nodes added to a bandwidth reserved stream are compliant with this amendment (each node reports its standard's version to the DM which then broadcasts this information to the domain, see clause 8.6.4).

DLL multicast stream establishment with BW reservation includes the same steps as specified in the previous clause.

In addition to these steps, the actions specified in the following clauses enable the nodes to reserve BW from the DM.

8.17.2.1 Bandwidth reservation when direct link from originating node to client node

Once the originating node receives the DMC_Path.cnf message from the DM, it shall decide whether bandwidth reservation is required for the stream. If bandwidth reservation is required, the client node of the DLL multicast stream shall request the DM for BW reservation.

If BW reservation from the DM is required, the originating node will ask the DM to reserve BW for the stream by sending a DMC_BWReserve.req message. The DMC_BWReserve.req message will include the BW required for the DLL multicast stream, and the rate from the originating node to the client node. The rate enables the domain master to estimate the time allocation needed to serve the DLL multicast transmission based on the number of bytes needed to be transmitted and the rate of the node.

Based on its calculations, the DM shall reply with a DMC_BWReserve.cnf message, and if bandwidth reservation was approved, the DM will then allocate TXOPs for this hop in the DLL multicast stream (identified by the MSID of the stream and the SID of the node) if this is the first bandwidth reservation for this DLL multicast stream, or change the allocation for existing TXOPs for this hop in the multicast stream if this is a bandwidth update for an existing stream.

8.17.2.2 Bandwidth reservation through relay

The source node will indicate that the DLL multicast stream -bandwidth reserved and that transmission should occur only in allocated TXOPs (this is done through the 'Is BW managed' field in the DMC_Path.req message).

A relay node that receives a DMC_Path.req message shall check whether this is a bandwidth reserved DLL multicast stream and act accordingly.

In addition to the steps described in clause 8.17.1 – DLL multicast stream establishment, the relay node shall send a DMC_BWReserve.req message to DM. The DMC_BWReserve.req message will include the SID of the relay node, the MSID of the stream, the DEVICE ID of the originating node of the stream and the requested bandwidth.

Based on this information the DM will calculate whether requested bandwidth can be allocated and reply with a DMC_BWReserve.cnf message. If bandwidth reservation was approved, the DM will then allocate resources to serve the hop in the DLL multicast stream.

8.17.32 Preventing loops and packets duplications

The paths of a specific DLL multicast stream shall be established in a tree topology that ensures that a node shall not receive duplicate multicast packets from different paths and prevent the source node or any relay node to duplicate unnecessarily transmissions. The topology of the DLL multicast stream tree is built under a principal rule that each node shall receive packets of a specific (OriginatingNode, MSID) only from one node. The DLL multicast stream paths tree shall be built according to this rule by executing the following procedure in path establishment: When a source node binds a new client node to an existing DLL multicast stream, it shall send towards it the DMC_Path.req message as defined in the previous ~~section~~clause. Any relay node on the path towards the newly joined client node shall verify that it always receives the DMC_Path.req for this specific (OriginatingNode, MSID) from the same sender node. In case that it receives a DMC_Path.req message from a node different from the sender node to which it is currently bound, it shall reply with the DMC_PathReject.cnf message towards the source node. The DMC_PathReject.cnf message shall contain the rejecting node's DEVICE_ID, the DEVICE_ID of the node that sent it DMC_Path.req message and the rejection reason (duplication source).

When the source multicast node receives the DMC_PathReject.cnf message, it may decide to release the entire tree or the branch and rebuild it again, or to enforce establishment of the path until the rejecting relay node based on the existing path. If the source node decides to enforce the existing path, it shall send the DMC_EnforcePath.req towards the relay node that encountered the problem via the original path. The source node shall address the DMC_EnforcePath.req message to the first relay node in the path toward the rejecting node. The DMC_EnforcePath.req message shall contain the full path until the rejecting node and the client node. Each relay node that receives the DMC_EnforcePath.req message shall forward the message to the next relay node according to the specified path toward the rejecting node. When the rejecting node receives the DMC_EnforcePath.req message it shall create a DMC_Path.req message, filling it with the information received in the DMC_EnforcePath.req message, and forward the message to the next relay node according to the current routing table. From this phase, the path establishment procedure towards the client node shall continue as specified in the previous ~~section~~clause. The client node shall reply with the DMC_Path.cnf message to the relay node that sent it the DMC_Path.req message. All the relay nodes on the path towards the source node upon receiving the DMC_Path.cnf message, shall execute the bind, update the Path_List field in the DMC_Path.cnf message with their own DEVICE_ID and forward the DMC_Path.cnf message towards the source node.

A specific relay that has to forward a specific MSID stream traffic to several nodes that are bound with this MSID may establish a PHY multicast group. In this case, the node may create or update PHY multicast groups when it receives a DMC_Path.req message to transmit the data to the next relay nodes or client nodes. In that case, the PHY multicast group shall only include bound client nodes and relay nodes that are in its bind list for this MSID in its current hop.

Node A decides to add node N to the '***MSID***' DLL multicast stream.
According to the current routing table node A sends the DMC_Path.req message Message to node D.

According to the current routing table node D sends the DMC_Path.req message to node R.

According to the current routing Relay node R sends the DMC_Path.req message To node E.

Node E knows that the legitimate source node for the specified '***MSID***' is node D, therefore it reject the DMC_Path.req message By sending the DMC_PathReject.cnf message to node R.

Node R receives the DMC_PathReject.cnf message and sends a DMC_PathReject.cnf message node D.

Node D receives the DMC_PathReject.cnf message and sends a DMC_PathReject.cnf message node A.

Node A sends the DMC_EnforcePath.req message To Node B to establish the path towards node N with a specified path until node E. Node B sends theDMC_EnforcePath.req message to node D (and not to node R) according to the specified path in the DMC_EnforcePath.req Message.

Node D sends the DMC_EnforcePath.req Message to node E and node E sends DMC_Path.req message to node N.

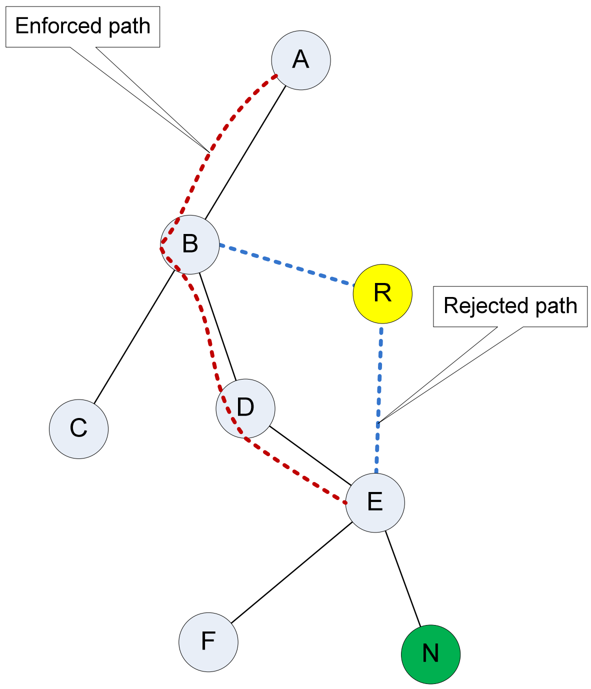

Figure 8-61.1 – Example of the mechanism for preventing loops in the DLL multicast tree

8.17.4~~3~~ Releasing client node from MSID

When the source node of a DLL multicast stream decides to release a client node from its MSID, it shall increment the Transaction ID of that DLL multicast stream and shall send a DMC_ReleasePath.req message to the respective client node or to its first relay node in case the client node is accessed via relay node(s). Each node that receives the DMC_ReleasePath.req shall release the specified node from this bind list and forward the message towards the client node. Each node that received the message shall reply with DMC_ReleasePath.cnf message to the node that sent it the DMC_ReleasePath.req message. Each relay node that does not have any nodes in its bind list shall released itself from the MSID multicast stream and indicate it in the replied DMC_ReleasePath.cnf message.

8.17.5~~4~~ Recovery from DLL multicast broken path

In case that one of the relay nodes determines that the path of a specific MSID is broken it shall inform the domain master via a normal topology update message and the source node of the DLL multicast stream via a DMC_BrokenLink.ind message.

The multicast source node may correct the broken path according to newly received updated routing table from the domain master. The source node may correct an existing path by sending DMC_ReleasePath.req to the relevant nodes and then it shall send new DMC_Path.req to the relevant nodes.

8.17.~~5~~6 Aging DLL multicast path process

In order to prevent a situation where a multicast source node leaves the network and all the respective nodes in the multicast stream path are still holding MSID resources, an aging mechanism shall be used. The source node of each DLL multicast stream (MSID) shall periodically send a management message: DMC_PathAlive.ind via the established MSID DLL multicast stream paths tree to the first node of each path (client node or relay node). Each node in the tree that receives this message shall reset its aging timer for that DLL multicast stream and shall transmit the DMC_PathAlive.ind message to each of the nodes that are bound to this DLL multicast stream, identified by (OriginatingNode, MSID), according to the binding information that it has configured during the DLL multicast stream paths establishment. Each node in the path that does not receive a DMC_PathAlive.ind message within a period of DMC_PATH_AGING_PERIOD (1 second) shall remove itself from this DLL multicast stream and release all of its MSID resources.

8.17.7 DLL Multicast stream bandwidth maintenance

Each node participating in a DLL multicast stream that is bandwidth reserved (whether it is the originating node or a relay node) is responsible for modifying its hop's requested bandwidth according to changes in the stream (e.g., if nodes are removed from a binding list, or if the hop's rate is changed, or if the hop transmission method is changed from using unicast connection to a PHY multicast connection).

If a node is no longer acting as a relay node in a DLL multicast stream, it is this node's responsibility to ask the DM to release all bandwidth reserved for its hop in the stream.

The modification of the bandwidth reservation is done by sending DMC_BWReserve.req to the DM. In this case, the DM shall reply with a DMC_BWReserve.cnf and change the allocation for the existing TXOPs for the indicated hop in the bandwidth reserved multicast stream.

The allocated bandwidth can be released by sending DMC_BWRelease.req to the DM. In this case, the DM shall reply with a DMC_BWRelease.cnf and completely remove the TXOPs assigned for the indicated hop in the bandwidth reserved multicast stream.

The internal rules used by the domain master to decide whether an allocation should be expanded or contracted due to ongoing flow maintenance done by the bandwidth management function are out of the scope of this Recommendation.

NOTE – If the bandwidth allocation for the DLL multicast stream is changed, this will be reflected in the MAP describing the following MAC cycles.

8.17.8 Transmission of bandwidth managed DLL Multicast stream

Originating node and relay nodes participating in a bandwidth managed DLL multicast streams should transmit the traffic associated with the multicast stream only in TXOPs assigned for that stream. This assignment is indicated via the 'Multicast Indication' flag,SID and MSID fields in the MAP's TXOP basic descriptor (see clause 8.8.4)

8.17.9~~6~~ DLL Multicast protocol messages

8.17.9.1~~6.1~~ DMC_Path.req message format

The format of the DMC_Path.req management message shall be as shown in Table 8-113.

Table 8-113 – Format of the MMPL of the DMC_Path.req message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL Multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
ClientID	2	[7:0]	The DEVICE_ID of the client node of the DLL Multicast steam source node.
MulticastAddress	3-8	[47:0]	MAC address of the multicast stream
Transaction_ID	9	[7:0]	Identifies this path transaction.
Relay ID	10	[7:0]	The DEVICE_ID of the relay node (NOTE 1)
Is BW Managed	11	[0]	0 – If this DLL multicast stream is not bandwidth reserved 1 – If this DLL multicast stream is bandwidth reserved (and transmission should occur only in allocated TXOPs) (NOTE 1)
Reserved	11	[7:1]	Reserved by ITU-T. (NOTE 2)
NOTE 1 - If there is a direct link between the source node and the client node, this field shall be set with the DEVICE_ID of the client node NOTE 2 - Bits that are reserved by ITU-T shall be set to zero by the transmitter and ignored by the receiver.			

8.17.9.26.2 DMC_Path.cnf message format

The format of the DMC_Path.cnf management message shall be as shown in Table 8-114.

Table 8-114 – Format of the MMPL of the DMC_Path.cnf message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL Multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
ClientID	2	[7:0]	The DEVICE_ID of the client node of the DLL Multicast steam source node.
Transaction_ID	3	[7:0]	Identifies this path establishment transaction. It shall contain the same value that was specified in the corresponding DMC_Path.req message.
NumOfNodes	4	[7:0]	Specifies the number of relay nodes (n) in the Path_List from the source node towards the client node.
Path_List[0]	5	[7:0]	This entry in the list contains the DEVICE_ID of the last relay node in the established path from the source node towards the client node.
Path_List[n-1]	4+n	[7:0]	This entry in the list contains the DEVICE_ID of the first relay in the established path from the source node towards the client node.

8.17.9.3~~6.3~~ DMC_PathReject.cnf message format

The format of the DMC_PathReject.cnf management message shall be as shown in Table 8-115.

Table 8-115 – Format of the MMPL of the DMC_PathReject.cnf message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL Multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
ClientID	2	[7:0]	The DEVICE_ID of the client node of the DLL Multicast steam source node
Transaction_ID	3	[7:0]	Identifies this path establishment transaction specified in the DMC_Path.req message.
RejectingNodeId	4	[7:0]	The DEVICE_ID of the relay node that rejects the DMC_Path.req message
Rejection_code	5	[7:0]	00_{16} – The request Path is conflicted because there is already a path established for the specified multicast stream with a different source node 01_{16} – The node is not able to support additional multicast streams 02_{16} to FF_{16} – reserved by ITU-T.

8.17.9.4~~6.4~~ DMC_EnforcePath.req message format

The format of the DMC_EnforcePath.req management message shall be as shown in Table 8-116.

Table 8-116 – Format of the MMPL of the DMC_EnforcePath.req message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL Multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
ClientID	2	[7:0]	The DEVICE_ID of the client node of the DLL Multicast steam source node.
MulticastAddress	3-8	[47:0]	MAC address of the multicast stream
Transaction_ID	9	[7:0]	Identifies this path establishment transaction. It shall contain the same value as in the original DMC_Path.req for this path.
NumOfNodes	10	[7:0]	Specifies the number of relay nodes (n) in the Path_List from the source node towards the rejecting node.
Path_List[0]	11	[7:0]	This entry in the list contains the DEVICE_ID of the first relay node in the established path from the source node towards the rejecting node.
Path_List[n-1]	10+n	[7:0]	This entry in the list contains the DEVICE_ID of the last relay in the established path from the source node towards the rejecting node

8.17.9.56.5 DMC_ReleasePath.req message format

The format of the DMC_ReleasePath.req management message shall be as shown in Table 8-117.

Table 8-117 – Format of the MMPL of the DMC_ReleasePath.req message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
ClientID	2	[7:0]	The DEVICE_ID of the client node of the DLL multicast steam source node to be release from the path.
Transaction_ID	3	[7:0]	Identifies this path transaction.
NumOfNodes	4	[7:0]	Specifies the number of relay nodes (n) in the Path_List from the source node towards the client node.
Path_List[0]	5	[7:0]	This entry in the list contains the DEVICE_ID of the first relay node in the established path from the source node towards the client node.
Path_List[n-1]	4+n	[7:0]	This entry in the list contains the DEVICE_ID of the last relay node in the established path from the source node towards the client node.

8.17.9.6~~6.6~~ DMC_ReleasePath.cnf message format

The format of the DMC_ReleasePath.cnf management message shall be as shown in Table 8-118.

Table 8-118 – Format of the MMPL of the DMC_ReleasePath.cnf message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
ClientID	2	[7:0]	The DEVICE_ID of the client node of the DLL multicast steam source node to be release from the path.
Transaction_ID	3	[7:0]	Identifies this path establishment transaction. It shall contain the same value as in the corresponding DMC_ReleasePath.req for this path.
RelayNodeStatus	4	[7:0]	Specifies the status of the relay node that sends this message. 0: The relay node released itself from the specified DLL multicast stream 1: The relay node still belongs to the specified DLL multicast stream 2 to 255: Reserved by ITU-T

8.17.9.76.7 DMC_PathAlive.ind message format

The format of the DMC_PathAlive.ind management message shall be as shown in Table 8-119.

Table 8-119 – Format of the MMPL of the DMC_PathAlive.ind message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL Multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.

8.17.9.86.8 DMC_BrokenLink.ind message format

This message is sent by a node that needs to report to the source of a DLL multicast stream that a link towards a multicast client node is broken.

The format of the DMC_BrokenLink.ind message shall be as shown in Table 8-120.

Table 8-120 – Format of the MMPL of the DMC_BrokenLink.ind message

Field	Octet	Bits	Description
Source ID	0	[7:0]	The DEVICE_ID of the source node of the DLL Multicast stream.
Reporting_DeviceID	1	[7:0]	The DEVICE_ID of the node reporting the broken link.
Broken_DeviceID	2	[7:0]	DEVICE_ID of the node with which the link is broken
StatusCode	3	[7:0]	0: the reporting node experienced a broken link 1: the reporting node has no bind information for this MSID 2 to 255: Reserved by ITU-T
NumberAffectedMSID	4	[7:0]	Number n of MSIDs affected by the broken link
MSID0	5	[7:0]	The multicast identification of the first affected MSID.
…	…	…	…
MSIDn	variable	[7:0]	The multicast identification of the n^{th} affected MSID.

8.17.9.9 DMC_BWReserve.req message format

The format of the DMC_BWReserve.req management message shall be as shown in Table 8-121

Table 8-121 – Format of the MMPL of the DMC_BWReserve.req message

Field	**Octet**	**Bits**	**Description**
Originating Node	0	[7:0]	The DEVICE_ID of the originating node of the DLL Multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
Requested BW	6-3	[39:8]	Specifies the requested bandwidth rate in bit/s, represented as a 32-bit unsigned integer.
Rate	8-7	[55:40]	The PHY data rate in bits per second for this hop in steps of 32 kbit/s…

8.17.9.10 DMC_BWReserve.cnf message format

The format of the DMC_BWReserve.cnf management message shall be as shown in Table 8-122

Table 8-122 – Format of the MMPL of the DMC_BWReserve.cnf message

Field	**Octet**	**Bits**	**Description**
Return Code	0	[7:0]	00_{16} – The requested bandwidth is approved. 01_{16} – The requested bandwidth is denied. 02_{16} to FF_{16} – reserved by ITU-
Originating Node	1	[7:0	The DEVICE_ID of the originating node of the DLL Multicast stream.
MSID	2	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.

8.17.9.11 DMC_BWRelease.req message format

The format of the DMC_ BWRelease.req management message shall be as shown in Table 8-123

Table 8-123 – Format of the MMPL of the DMC_ BWRelease.req message

Field	Octet	Bits	Description
Originating Node	0	[7:0]	The DEVICE_ID of the originating node of the DLL Multicast stream.
MSID	1	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.

8.17.9.12 DMC_ BWRelease.cnf message format

The format of the DMC_ BWRelease.cnf management message shall be as shown in Table 8-124

Table 8-124 – Format of the MMPL of the DMC_ BWRelease.cnf message

Field	Octet	Bits	Description
MSID	0	[7:0]	The multicast identification allocated by the source of the DLL multicast stream.
Originating Node	1	[7:0]	The DEVICE_ID of the originating node of the DLL Multicast stream.

9 Add text for new clause 8.20 "Metrics acquisition" as follows:

8.20 Metrics acquisition

The goal of metrics acquisition mechanism is to provide a node a way to calculate the throughput metrics between it and a destination node without needing to establish a flow.

A node that wants to obtain the achievable throughput with another node of the domain may request metric information to the DM by sending the message MA_AcquireMetrics.req, which includes the information on the destination node.

Upon reception of MA_AcquireMetrics.req, the DM shall respond to the requesting node with an MA_AcquireMetrics.cnf message.

8.20.1 Metrics acquisition protocol messages

8.20.1.1 Format of MA_AcquireMetrics.req

This message is sent from a node requesting an update of metrics information to the DM.

The format of the MMPL of the MA_AcquireMetrics.req shall be as shown in Table 8-125.

Table 8-125 – Format of the MMPL of the MA_AcquireMetrics.req message

Field	Octet	Bits	Description
OriginDeviceID	0	[7:0]	DEVICE_ID of the node origin of the link for which the metrics are requested
DestinationDeviceID	1	[7:0]	DEVICE_ID of the node destination of the link for which the metrics are requested

8.20.1.2 Format of MA_AcquireMetrics.cnf

This message is sent by the DM to inform a requesting node about the metrics of a particular link.

The format of the MMPL of the MA_AcquireMetrics.cnf shall be as shown in Table 8-126.

Table 8-126 – Format of the MMPL of the MA_AcquireMetrics.cnf message

Field	Octet	Bits	Description
OriginDeviceID	0	[7:0]	DEVICE_ID of the node origin of the link for which the metrics are requested
DestinationDeviceID	1	[7:0]	DEVICE_ID of the node destination of the link for which the metrics are requested
MetricsRouteList	Variable	See Table 8-127	Routing list from OriginDeviceID toward DestinationDeviceID. This field is only present when there is at least one relay node ($N \geq 1$)
MaxBitsPerSecond	Variable	[15:0]	Maximum data rate in bits per second from OriginDeviceID to DestinationDeviceID. (Note 1)
AttainableBitsPerSecond	Variable	[15:0]	The attainable data rate in bits per second from OriginDeviceID to DestinationDeviceID that a DM can allocate for a flow between these two nodes under current domain traffic conditions.
NOTE 1: This value is calculated from the values reported to the DM by each of the nodes of the domain in the BitsPerSecond field of the Visibility_List field of the TM_NodeTopologyChange.ind message (See Table 8-48) using the following formula: $MaxBitsPerSecond = \frac{1}{\sum_{i=1}^{N+1} \frac{1}{BitsPerSecond_i}}$ where N is the number of relays between OriginDeviceID and DestinationDeviceID.			

Table 8-127 – Format of MetricsRouteList

Field	Octet	Bits	Description
NumRelays	0	[7:0]	Number of relay nodes (N) in the MetricsRouteList
RelayNode[1]	1	[7:0]	DEVICE_ID of the first relay node in the list
…	…	…	…
RelayNode[N]	N	[7:0]	DEVICE_ID of the Nth relay node in the list
LinkMetrics[1]	N + 1 and N + 2	[15:0]	PHY data rate in bits per second from the originating node in the route to the first relay. (Note 1)
….	…	…	…
LinkMetrics[N+1]	3*N + 1 and 3*N + 2	[15:0]	PHY data rate in bits per second from the Nth relay in the route to the destination node. (Note 1)
NOTE 1: This value corresponds to the value reported to the DM by each of the nodes of the domain in the BitsPerSecond field of the Visibility_List field of the TM_NodeTopologyChange.ind message. See Table 8-48			

10 Add new Annex X "Test vectors" as follows:

Annex X – Test vectors

This annex includes test vectors for core operations described in this Recommendation.

X.1 CCM encryption

This clause provides a set of test vectors for parameters involved in CCM encryption described in clause 9.1. Parameters are expressed in a hexadecimal form with the leftmost byte representing the lowest byte within a parameter (i.e., byte 0). Within a byte, the leftmost bit represents the MSB.

X.1.1 CCM test vector 1

X.1.1.1 Input parameters

This clause provides one set of examples for input parameters used in CCM encryption.

Data packet, APDU (75 bytes):

The APDU can be broken into the following parameters:

Destination MAC address, *DA* (6 bytes)

DA = 00 00 5E 10 20 09_{16}

Source MAC address, *SA* (6 bytes)

SA = 00 00 5E 07 20 13_{16}

MAC client length/type, *LT* (2 bytes)

LT = 08 00_{16}

APDU payload, *P* (57 bytes)

```
P =      52 41 56 49 3B 45 52 45 5A 3B 41 56 4E 45 52 3B 4C 45 53 3B 52
         4F 59 3B 4D 41 52 43 4F 53 3B 41 47 55 53 54 49 4E 3B 4A 4F 48
         4E 3B 4A 42 3B 54 4F 4E 47 3B 56 5A 45 49 42
```
$_{16}$

Frame check sequence, *FCS* (4 bytes)

FCS = A4 55 5A 26$_{16}$

Note that VLAN TAG does not exist in this example (i.e., *TG* = 0), *Alen* is 21 bytes (see Table 9-5), and *Plen* is 57 bytes.

Parameters for the LLC frame header (LFH) are given as LLCFT = 2 (APDU), TSMPI = 0 (TSMP field not present), CCMPI = 1, LPRI = 0, FLEN = 71, MCSTI = 0, OriginatingNode = 1, DestinationNode = 2, BRCTI = 0, and TTL = 0 (see clause 8.1.3.1.1).

Resulting LFH (6 bytes):

LFH = 12 47 00 01 02 00$_{16}$

Parameters for the CCMP header (CCMPH) are given as MIC size = 111_2 (16-byte MIC), NN or NMK, Key ID = 0, and FN = 1 (see clause 9.1.2.3).

Resulting CCMP header, CCMPH (6 bytes):

CCMPH = 07 01 00 00 00 00$_{16}$

Encryption Key K: 47 68 6F 43 65 72 74 66 32 30 31 33 47 68 6E 43$_{16}$.

X.1.1.2 Parameters generated

This clause provides one set of examples of parameters generated by the node based on given set of input parameters.

Nonce, *N* (13 bytes):

N = 00 00 00 5E 07 20 13 00 00 00 00 00 01$_{16}$

Nonce block, B_0 (16 bytes):

B_0 = 79 00 00 00 5E 07 20 13 00 00 00 00 00 01 00 39$_{16}$

Associated data block, B_1 (16 bytes):

B_1 = 00 15 00 00 00 00 5E 10 20 09 00 00 5E 07 20 13$_{16}$

Associated data block, B_2 (16 bytes):

B_2 = 12 47 00 01 02 08 00 00 00 00 00 00 00 00 00 00$_{16}$

Payload block, B_3 (16 bytes):

B_3 = 52 41 56 49 3B 45 52 45 5A 3B 41 56 4E 45 52 3B$_{16}$

Payload block, B_4 (16 bytes):

B_4 = 4C 45 53 3B 52 4F 59 3B 4D 41 52 43 4F 53 3B 41$_{16}$

Payload block, B_5 (16 bytes):

B_5 = 47 55 53 54 49 4E 3B 4A 4F 48 4E 3B 4A 42 3B 54$_{16}$

Payload block, B_6 (16 bytes):

B_6 = 4F 4E 47 3B 56 5A 45 49 42 00 00 00 00 00 00 00$_{16}$

Counter block 0, Ctr_0 (16 bytes):

Ctr_0 = 01 00 00 00 5E 07 20 13 00 00 00 00 00 01 00 00$_{16}$

Counter block 1, Ctr_1 (16 bytes):

```
Ctr1 =   01 00 00 00 5E 07 20 13 00 00 00 00 00 01 00 01₁₆
```

Counter block 2, Ctr_2 (16 bytes):

```
Ctr2 =   01 00 00 00 5E 07 20 13 00 00 00 00 00 01 00 02₁₆
```

Counter block 3, Ctr_3 (16 bytes):

```
Ctr3 =   01 00 00 00 5E 07 20 13 00 00 00 00 00 01 00 03₁₆
```

Counter block 4, Ctr_4 (16 bytes):

```
Ctr4 =   01 00 00 00 5E 07 20 13 00 00 00 00 00 01 00 04₁₆
```

Cyphertext

```
C =        04 4f 24 21 07 fb 58 68 ba 1a c7 c3 1f 5c e7 20 c1 a2 09 ed a0
           29 d5 03 b1 e0 94 43 ed 4f 28 24 62 c8 28 c5 53 50 95 74 86 fc
           ea 0e 92 4d 2c 4f 3b 25 cf b5 3a 5e 1f 5e 3f₁₆
```

Message Integrity code

```
MIC =      8c df 6e 79 03 0f 7e 69 cc 33 b8 29 ef e4 6d e2₁₆
```

X.2 PAK test vectors

This clause provides a set of test vectors for parameters involved in the PAK protocol described in clause 9.2.2. Parameters are expressed in a hexadecimal form with the leftmost bit representing the MSB.

X.2.1 PAK test vector 1

X.2.1.1 Input parameters

This clause provides one set of examples for input parameters known to the supplicant and the authenticator before the initiation of key authentication process.

Node identifier of the supplicant, *A* (48 bits, see clause 9.2.2.2.1):

```
A =       0019 A717 DD30₁₆
```

Node identifier of the authenticator, *B* (48 bits, see clause 9.2.2.2.1):

```
B =       0019 A770 8A32₁₆
```

Node password shared by the supplicant and the authenticator, *PW* (96 bits, see clause 9.2.2.2.2):

```
PW =      5962 A05A B89F C0AA FB14 0EF7₁₆
```

X.2.1.2 Parameters generated or exchanged

This clause provides one set of examples of parameters generated and/or exchanged by the supplicant and the authenticator.

Secret exponent generated by the supplicant, *RA* (384 bits, see clause 9.2.2.2.5):

```
RA =      89A1 A7B4 F433 9220 2C60 960D 172A 7C45 6B95 C225 26B1 1C7A
          9E2E 7712 C43C 9C77 48B6 3936 A62B CF90 3C03 A0E2 0E28 D660₁₆
```

Secret exponent generated by the authenticator, *RB* (384 bits, see clause 9.2.2.2.5):

```
RB =      F052 57CB 1840 6A91 173B 87E4 1F22 9289 7D3E 08A7 BCA0 4EB9
          1A8A CFF3 940C AE00 E15B 302D 7E67 2E81 CCB4 C103 A241 B133₁₆
```

Concatenated input parameters, $P = A \mid B \mid PW$ (192 bits, see [ITU-T X.1035]):

P =	0019 A717 DD30 0019 A770 8A32 5962 A05A B89F C0AA FB14 0EF7$_{16}$

Intermediate result, $IR1 = H_1(P)$ (1152 bits, see clause 9.2.2.2.6 and [ITU-T X.1035]):

IR1 =	11F9 E6DD 6E7D 48EF 3672 CA0F A2EC 2488 7678 34B9 506C FE86
	5BC0 A847 3051 F6FD 408D 0178 816D 80A7 D8D3 B75C 3176 C8D3
	BD12 2AD1 2AE5 C26C 29F8 3518 BD91 1581 9483 C303 68F3 B137
	3A33 A8E5 6193 83B8 34F1 59B4 E1C3 8259 B3DA D35F 7876 A7FE
	3B0A 9E9A F594 BEA6 B126 77B0 50EC 672E 11F7 3A1E 231E 9ECE
	793A 34AE 154D 4EB0 82BB AC26 1F8E 0B50 735C 01FB C364 9081$_{16}$

Intermediate result, $IR2 = g^{RA}\ mod\ p$ (1024 bits, see clause 9.2.2.2.3, clause 9.2.2.2.4, and [ITU-T X.1035]):

IR2 =	D678 B9D6 E866 FB46 4865 A430 C2BA 0668 722D 236E 7BEA 1C51
	7E4A 4812 1CD4 B42C 7803 2B8C F05F 497B 46EC F894 CB5A 0678
	7104 7E99 448A D384 46A1 15AF 4640 7B9B F13C FFBD 2452 FB69
	3D7C 6445 DE1E 95AF DC13 7B33 01AE 6659 0839 A05E 03A2 2169
	E10C C5F6 D87B 62E5 FF92 B000 4DA9 8058 9F95 5F2E F66A 42D6
	CBC4 E70A A3CA D136$_{16}$

Parameter carried in AUT_NodeRequest.req, $X = IR1 \cdot IR2$ (2176 bits, see clause 9.2.5.1.1 and [ITU-T X.1035]):

X =	0F0F 612E 0137 3C14 AB36 88FB 07C9 98E6 EBA7 033C E635 4EDA
	54D2 DA67 46D2 43AC FC19 3F9E 7E66 4B5F 1ED8 13D7 7763 0BFF
	DE60 E3D5 397E 901A 1338 99CC 2E52 209E 441F 0DDE 9449 1CDA
	8B36 B454 FF1B 1E9E 784A 07D4 5DF5 85C5 503E 65AD 7E34 EE82
	2E92 99AC B766 EF21 0CEB 7D10 B620 AB10 BA09 7DF7 EEB0 25BE
	E6AD 223B 3049 93F9 FCDB C996 EA09 8BFC 56C7 495E 2E17 BD88
	E201 B2C2 40E9 9F79 B681 9963 3D8F 5F22 7BD8 5373 A868 902D
	93FC 20CB 9F1D 369B 1C54 A143 E416 D7C5 2A59 EAC8 0B49 D013
	575F C302 FA4D AD02 DDF7 BA96 71E9 9B56 DE44 9E57 9DFB 83AD
	B1F1 1A43 0900 2F9C 8EFD A771 0A71 DAA0 176D E5ED C7A8 02F3
	99D8 6E26 0458 3EF1 901F 7C1A 99E8 CBB7 5357 09DA 84F2 5393
	9F2E 3706 79F9 CC36$_{16}$

Intermediate result, $IR3 = H_2(P)$ (1152 bits, see clause 9.2.2.2.6 and [ITU-T X.1035]):

IR3 =	2773 D699 51BB 3CC5 D595 F28E 3AAF CCBF C2A3 895D D429 A707
	13EE C1D7 2E08 BCA9 D3C7 AE45 7317 5180 25AE 9B9D 6125 BED6
	EA69 F440 FD1F D309 2404 0AD9 E3DB B2A4 8F1A 49DA 0F14 BD2E
	15B7 2E9D E16E 9E95 EE26 6890 AA45 1ACE A1A7 394C 9BFB 55B8
	54DE 5CFB 1385 028D 3A58 ED53 C8B1 639C 76D4 F4AF BB51 52D8
	2E7F F099 4210 DA52 CDFD DF2B 973D EC89 DFEB A32C A4B7 4428$_{16}$

Intermediate result, *IR4* = g^{RB} *mod p* (1024 bits, see clause 9.2.2.2.3, clause 9.2.2.2.4, and [ITU-T X.1035]):

IR4 =	B503 D0FE AAC7 D9D5 B2C1 ADAC ACB2 F4AC ED7D E0EA 65F2 D88F
	39DA A98C CCA3 C197 40F4 B466 6DC4 310F 6969 482F 2B94 D5A2
	BB64 4E8F 04A7 12D4 81FF 34E0 45F3 E351 E255 3A57 F32F E600
	820A 7C9B 0407 F35C 588D 4C6A 0908 BD7C 9F76 A9BB A478 16BB
	C6F8 73DC B9EF C0CD 54FC E949 F840 2EE6 DD0C D4B9 52F9 96BC
	D529 9885 964C 394D$_{16}$

Parameter carried in AUT_Promp.ind, *Y* = *IR3·IR4* (2176 bits, see clause 9.2.5.1.2 and [ITU-T X.1035]):

Y =	1BE5 7D4B 4832 8ED0 90C6 5623 C4D1 1400 F58E FDEF A37E 2AAC
	0EB6 9A5E 904E F71A 193D 46B2 6113 372A 0517 45CB 1FCB 5200
	2FF9 A00C 9070 72C2 5946 E87D 630E 36A2 AFCE 5FB4 AA35 D2F7
	74DA FED9 11A4 4EF4 698C 4582 9E47 8AA6 F74A 6714 09E0 8CA3
	9654 1D65 9099 DC16 3C40 2E8D 6779 D9CD B182 4EF6 6A83 8A40
	7537 64A9 ABDB 1619 33A7 44E6 8C9A 3D37 2D34 1C46 423F 4679
	B03B 563D 0B02 D397 6171 776F 7FD4 31D6 6B26 4F6A 5AC5 BD89
	434C C914 2698 36A0 DC88 2E31 D3FD B108 69FA 4F86 AFAB CDDC
	7CE6 D753 F7C5 5286 6E12 C2C3 80E4 70A2 6F81 08CD D379 08F5
	AE54 4467 DA86 974D BE27 39A6 5058 E201 1387 AB08 6402 15A3
	E973 5002 8852 6DD8 302B 60F8 28AF 9806 4535 F825 425C 0652
	010F 763A 052C 6808$_{16}$

Intermediate result, *IR5* = *P* \| g^{RA} *mod p* \| g^{RB} *mod p* \| $g^{RA·RB}$ *mod p* (3264 bits, see clause 9.2.2.2.3, clause 9.2.2.2.4, and [ITU-T X.1035]):

IR5 =	0019 A717 DD30 0019 A770 8A32 5962 A05A B89F C0AA FB14 0EF7
	D678 B9D6 E866 FB46 4865 A430 C2BA 0668 722D 236E 7BEA 1C51
	7E4A 4812 1CD4 B42C 7803 2B8C F05F 497B 46EC F894 CB5A 0678
	7104 7E99 448A D384 46A1 15AF 4640 7B9B F13C FFBD 2452 FB69
	3D7C 6445 DE1E 95AF DC13 7B33 01AE 6659 0839 A05E 03A2 2169
	E10C C5F6 D87B 62E5 FF92 B000 4DA9 8058 9F95 5F2E F66A 42D6
	CBC4 E70A A3CA D136 B503 D0FE AAC7 D9D5 B2C1 ADAC ACB2 F4AC
	ED7D E0EA 65F2 D88F 39DA A98C CCA3 C197 40F4 B466 6DC4 310F
	6969 482F 2B94 D5A2 BB64 4E8F 04A7 12D4 81FF 34E0 45F3 E351
	E255 3A57 F32F E600 820A 7C9B 0407 F35C 588D 4C6A 0908 BD7C
	9F76 A9BB A478 16BB C6F8 73DC B9EF C0CD 54FC E949 F840 2EE6
	DD0C D4B9 52F9 96BC D529 9885 964C 394D 9768 81A5 5808 E976
	F569 319A 8764 8539 16E0 1496 6E1F 191A 482B 1838 0E4F 9A77
	99FA C4AF AE0B 9C74 7A57 630C DA71 DF19 5CB2 FE5F B951 52B7

```
              EADB C460 8B62 3464 944E 1011 8471 028C 8000 8F8E EC8E B6C7
              FC36 30DF 27DD 2D43 3277 2FB4 E1A8 FF9F CA61 6E4E E466 CDA4
              B6AD 9B02 F498 39BF 589B C793 2680 8C26 9AA6 B351 9418 EFEB16
```

Parameter carried in AUT_Promp.ind, $S_1 = H_3(IR5)$ (128 bits, see clause 9.2.2.2.6, clause 9.2.5.1.2, and [ITU-T X.1035]):

S_1 = 3BB5 5C57 33CF 1E7F 0711 C525 CD89 3181_{16}

Parameter carried in AUT_Verification.res, $S_2 = H_4(IR5)$ (128 bits, see clause 9.2.2.2.6, clause 9.2.5.1.3, and [ITU-T X.1035]):

S_2 = 3DB1 A72C 64B0 CAE6 57FF D4EA DC31 $F676_{16}$

NSC key generated, $K = H_5(IR5)$ (128 bits, see clause 9.2.2.2.6 and [ITU-T X.1035])

K = ABA6 D8E8 BD2B 705B B4CC 34BD 1107 $E00D_{16}$