Project	***IEEE 802.16j Mobile Multihop Relay Task Group***
Title	***An Advanced ARQ Scheme (A^{2}RQ) on Relay Link for 802.16j***
Date	***2006-11-07***
Source(s)	**Toshiyuki Kuze, Shigeru Uchida, Kentaro Sawa,** *Mitsubishi Electric Corp* *5-1-1 Ofuna Kamakura, Kanagawa* *2478501, Japan* Voice: +81-467-41-2885 Fax: +81-467-41-2486 ***Kuze.Toshiyuki@ah.MitsubishiElectric.co.jp*** *Uchida.Shigeru@dh.MitsubishiElectric.co.jp* *Sawa.Kentaro@bk.MitsubishiElectric.co.jp* **Jeffrey Z. Tao, Koon Hoo Teo, Jinyun Zhang** *Mitsubishi Electric Research Lab* *201 Broadway* *Cambridge, MA 02139 USA* Voice: 617-621-{7557,7527} Fax: 617-621-7550 *{tao, teo, jzhang}@merl.com*
Re:	*Response to the call for proposal of the 802.16j relay TG (i.e., IEEE 802.16j-06/027, "Call for Technical Proposals regarding IEEE Project P802.16j", October 15, 2006).*
Abstract	*This contribution describes an enhancement to the current IEEE 802.16e ARQ mechanism.*
Purpose	*To enhance the current IEEE 802.16e ARQ mechanism as suggested in this contribution.*
Notice	
Release	
Patent Policy and Procedures	

An Advanced ARQ Scheme (A^2RQ) on Relay Link for 802.16j

An Advanced ARQ Scheme (A^{2}RQ) on Relay Link for 802.16j

***Toshiyuki Kuze**, Shigeru Uchida, Kentaro Sawa*
Mitsubishi Electric Corp
5-1-1 Ofuna Kamakura, Kanagawa
2478501, Japan

Jeffrey Z. Tao, Koon Hoo Teo, Jinyun Zhang
Mitsubishi Electric Research Lab
201 Broadway
Cambridge, MA 02139 USA

1. Introduction

IEEE 802.16j MMR imposes a demanding performance requirement on relay stations, which will functionally serve as an aggregating point on behalf of the BS for traffic collection from and distribution to the multiple MSs associated with them. The OFDMA-based 802.16e standard [2], however, is designed and optimized solely for single-hop communication between SSs/MSs and a BS in the PMP mode. As a result, its direct application on the relay link may render a potential bottleneck and preponderantly limit the overall network performance. In particular, when the ARQ protocol is deployed directly on the high speed relay link, the severity of window lock effect and undesirable sensitivity to CINR estimation will be exacerbated, ultimately leading to dismal performance degradation. As a solution, we propose an enhanced ARQ mechanism to accomplish reliability, high capacity, low delay and jitter on the relay link, which is essential to the success of multihop relay network. The enhanced ARQ scheme, which is called A^{2}RQ, mitigates the window lock effect, lessens the sensitivity to CINR estimation and leverages the coding gain at MAC layer, thereby delivering a performance superior to the current IEEE 802.16e ARQ scheme. In the following, a brief summary of the proposed A^{2}RQ scheme is provided in Section 2, while the performance evaluation results are discussed in Section 3. Section 4 completes this contribution with detailed proposed text change.

2. Summary of Proposal

The system operation, namely the operations at transmitter and receiver side, is illustrated in Figure 1 and Figure 2**,** respectively; and can be summarized as follows.

2.1 Transmitter side

The operations at transmitter side can be divided into following steps:

- ***Concatenation***:
  - Multiple MSDUs can be concatenated to create a long MSDU, with a concatenation header being placed in front of each individual MSDU to indicate its length. Note that MSDUs of different CIDs may be concatenated together, which essentially implies an operation of connection aggregation. To maintain QoS, only CIDs corresponding to the same service quality level can be aggregated. The actual algorithm to select connections to be aggregated is implementation dependent. Padding bits may be needed in order for total length of the resultant concatenated MSDU to meet certain requirement.

- ***Encoding:***
  - The concatenated MSDU then will be encoded by an erasure correction code (ECC). As a result, multiple system packets and parity packets of equal size will be generated.

- ***Final frame construction and transmission:***

- Sequence number will be assigned to each coded packet. In addition, group sequence number will be used to identify all the coded packets that are originated from the same concatenated MSDU prior to encoding.
- For error detection, CRC field will be inserted into the coded packets train in a periodic manner.
- The coded packet train and the associated CRC fields will be fragmented into a few fragments. A MAC header will be attached in front of each fragment for transmission.

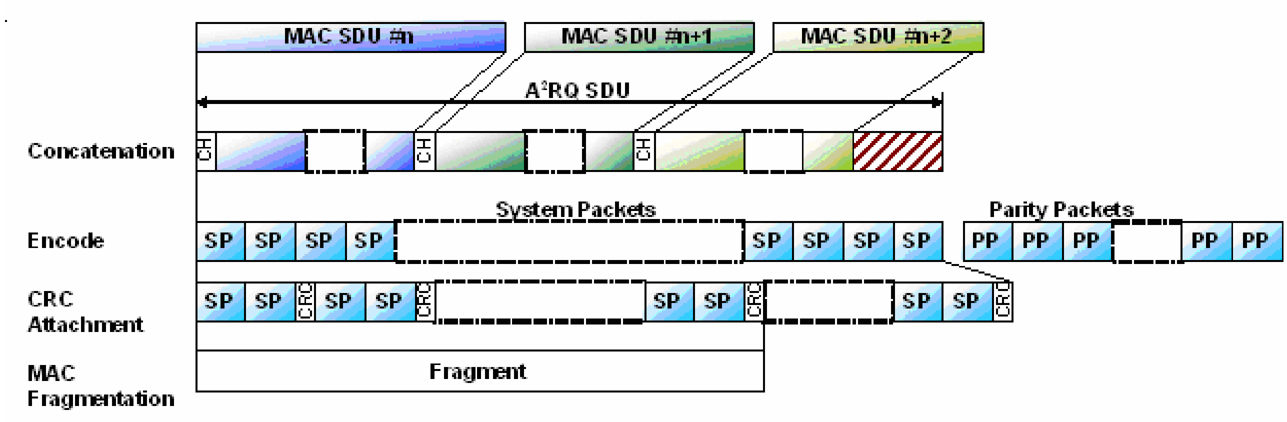

Figure 1: System operations at transmitter side.

2.2 Receiver side

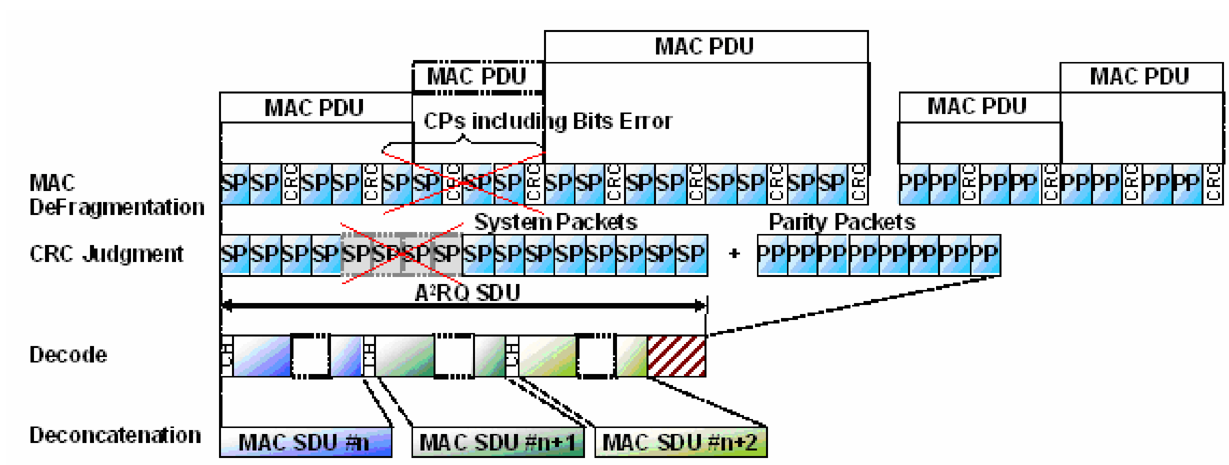

Figure 2: System operations at receiver side.

Similarly, the operations at receiver side can be divided into following steps:

- ***Decoding:***
  - Upon reception, consecutive MPDUs will be assembled together. The coded packets and CRC fields will be located, and CRC fields will be used to detect any error. If no error is detected. An attempt to decode the coded data packets (a.k.a. system packets) will be made, and the transmitter will be notified of the status of decoding (i.e., success or failure). The transmitter

determines whether or not to transmit more parity packets further in the sequence based upon the feedback.

- ***De-concatenation:***
  - Once the original concatenated MSDU is recovered, all the appended concatenation headers will be removed, and individual MSDU contained therein will be retrieved and delivered to the higher (sub)layer.

3. Performance Results

To evaluate the performance of the proposed enhanced ARQ (A^2RQ) protocol, a system level simulation platform has been developed, which accurately models all the protocol details of OFDMA PHY and MAC defined in IEEE 802.16e. Table 1 and Figure 3 describe the settings of the simulation.

Table 1: Simulation Settings

	Parameter	**Value**	
	Channel Model	*SUI*	
	Path loss Model	*Cost 231*	*NLOS*
	Shadowing	*Log normal Shadowing*	
	Cell Number	*19 cells*	
	Number of user	*1*	*Center cell*
	AMC	*1. QPSK R=1/2* *2. 16QAM R=1/2* *3. 64QAM R=1/2* *4. 64QAM R = 3/4*	*Please refer to Figure 4*
	Center Frequency [GHz]	*2.6*	
	Bandwidth [MHz]	*10*	
	BS Tx Power [W]	*10*	
	Mobility	*Fixed* *Pedestrian* *Vehicular*	*0km/h* *4km/h* *40km/h*
	Required BER	10^{-3}	
	Maximum MAC PDU Size [Bytes]	*2047*	
	ACK Period [frame]	*10*	
A^2RQ	ARQ Window Size	*1024*	
	ARQ Block Size	*128*	
	Max CP Size [Byte]	*128*	
	Num of Initial PP	*0*	*First transmission attempt*
	Num of additional PP	*9*	*Subsequent transmission contingent upon the failure of initial transmission*
	MAC Scheduler	*Proportional fairness*	
	Traffic Load Factor[%]	*100,66,50,33,25*	
	Simulation time	*5100 frames*	

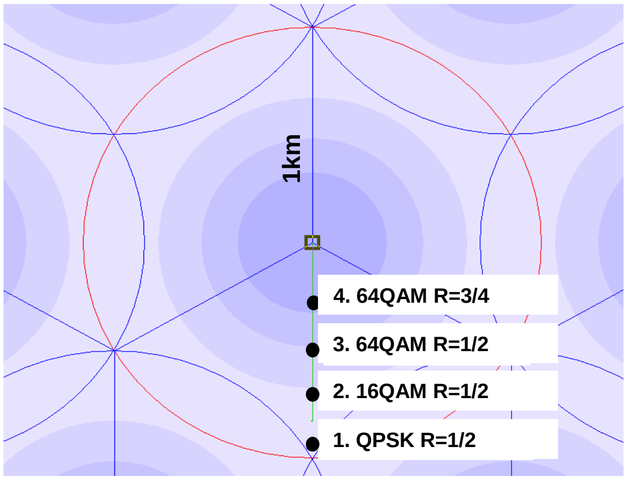

Figure 3: AMC settings

Figure 4 compares the throughput performance of proposed A^2RQ and general ARQ (G-ARQ) scheme of 802.16e. The figure evidently demonstrates that as the traffic load increases, the throughput of G-ARQ enters a plateau of 5 Mbps, while A^2RQ can sustain a throughput as high as 10 Mbps. This significant performance improvement achieved by A^2RQ is primarily due to its capability of mitigating the window lock effect that plagues the G-ARQ in heavy traffic condition.

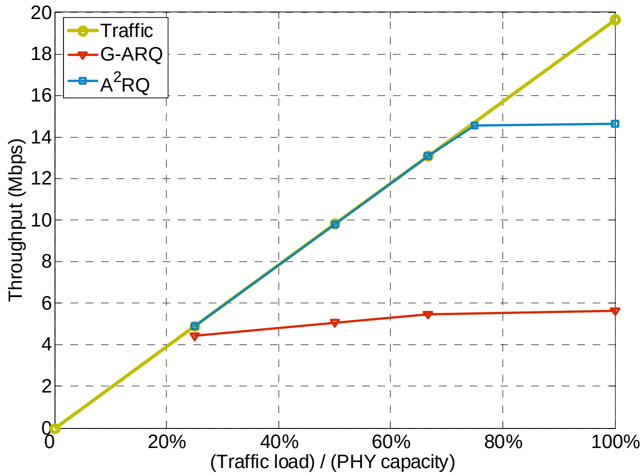

Figure 4: Throughput comparison: A^2RQ versus G-ARQ (64QAM R=3/4, buffer =5580 blocks)

Delay is another key statistics that deserves a close examination. Figure 5 and Figure 6 depict the mean delay and jitter for both A^2RQ and G-ARQ, respectively. As compared to the G-ARQ, these two figures confirm that A^2RQ excels in its delay performance as well.

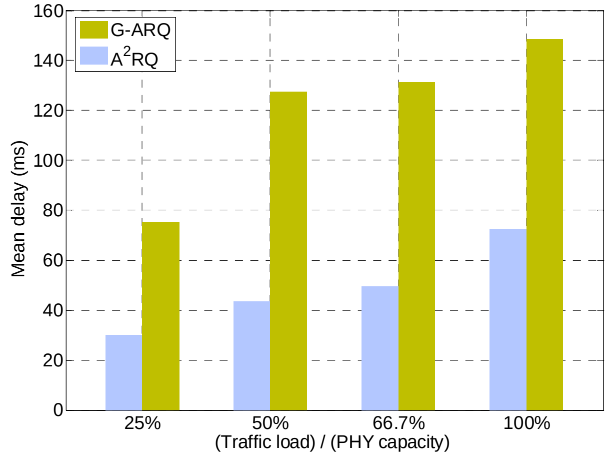

Figure 5: Mean delay comparison: A^{2}RQ versus G-ARQ (64QAM R=3/4, buffer = 1024 blocks).

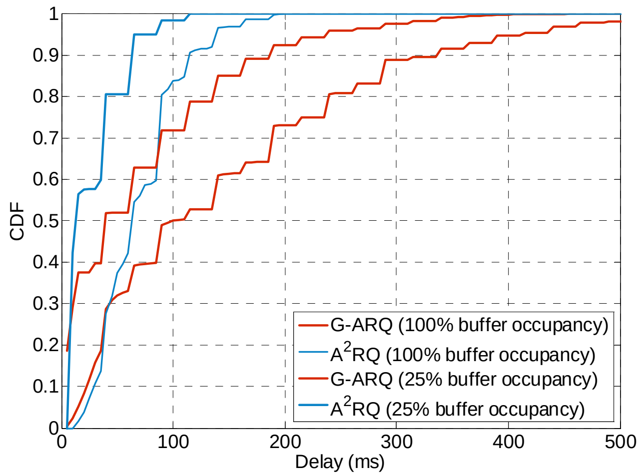

Figure 6: Delay distribution comparison: A^{2}RQ versus G-ARQ (64QAM R=3/4, buffer = 1024 blocks).

IEEE 802.16e system relies on proper feedback of CINR estimation to perform AMC function. It has been found that G-ARQ is very sensitive to the CINR estimation value. For example, Figure 7 and Figure 8 show that a 3dB overestimation can easily cost G-ARQ a dismal 75% performance degradation in a pedestrian mobility environment. On the other hand, due to the coding gain it enjoys, A^{2}RQ can dampen this undesirable sensitivity, and thus can still maintain a superior throughput even with an inaccurate CINR estimation.

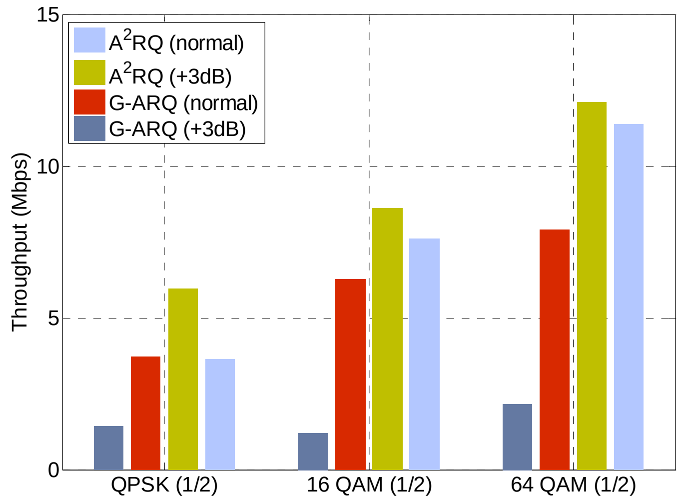

Figure 7: Sensitivity to CINR estimation - AMC.

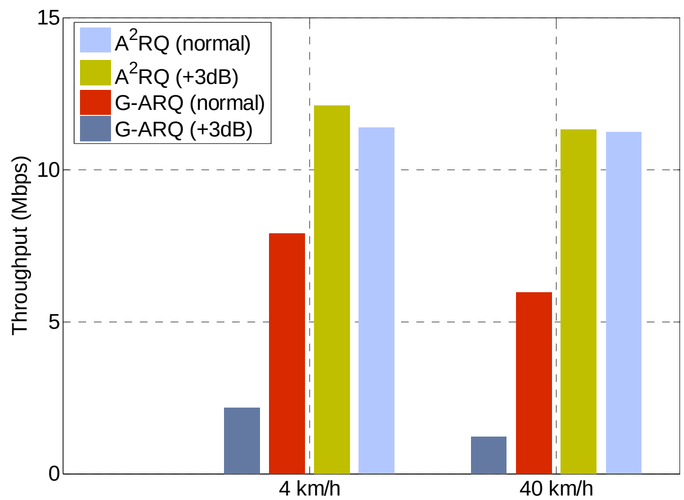

Figure 8: Sensitivity to CINR estimation - mobility.

4. Proposed Text Changes

6. MAC common part sublayer

6.3.2 MAC PDU formats

6.3.2.1.1 Generic MAC header

Change the text in Table 6 as indicated:

Table 6 – Type encodings

Type bit	Value
……	……
#4	*ARQ Feedback Payload and A^2RQ Feedback Payload*

	1 = present, 0 = absent
#3	*……* *For ARQ-enabled connections, this bit shall be set to 1.* *For A^2RQ-enabled connections, this bit shall also be set to 1.*
……	*……*

6.3.2.2 MAC subheader and special payloads

Insert the following paragraph at the end of 6.3.2.2.

Various types of subheaders may be present in a MAC PDU with generic MAC header. For a connection that enables A^2RQ, fragmentation subheader (FSH) and/or packing subheader (PSH) will be used in the same manner as that for an ARQ-enabled connection. The A^2RQ subheader (A^2SH) shall be placed immediately after FSH and PSH. The FSH and PSH contain the group sequence number (GSN), while the A^2RQ subheader contains the coded packet number (CPN) and coded packet (CP) length.

6.3.2.2.1 Fragmentation subheader

Change the text in Table 8 as indicated.

Table 8 – Fragmentation subheader format

Syntax	**Size**	**Notes**
Fragmentation Subheader ( ){		
FC	*2 bits*	*Indicates the fragmentation state of the payload* *00= no fragment* *01= last fragment* *10= first fragment* *11= continuing(middle)fragment*
if(ARQ-enabled Connection)		
BSN	*11 bits*	*Sequence number of first block in the current SDU fragment.*
else if (A^2RQ enabled Connection)		
GSN	*6 bits*	*Group Sequence number of first block in the Coded Packet*
reserved	*5 bits*	*-*
else {		
if (Type bit Extended Type)		*See Table 6*
FSN	*11 bits*	*Sequence number of the current SDU fragment. This field shall increment by one (modulo 2048) for each fragment, including unfragmented SDUs.*
Else		
FSN	*3 bits*	*Sequence number of the current SDU fragment.* *This field shall increment by one (modulo 8) for each fragment, including unfragmented SDUs.*
}		
Reserved	*3 bits*	
}		

6.3.2.2.3 Packing subheader

Change the text in Table 11 as indicated.

Table 11 – Packing subheader format

Syntax	Size	Notes
Packing subheader ( ){		
FC	*2 bits*	*Indicates the fragmentation state of the payload* *00= no fragment* *01= last fragment* *10= first fragment* *11= continuing(middle)fragment*
if(ARQ-enabled Connection)		
BSN	*11 bits*	*Sequence number of first block in the current SDU fragment.*
else if (A^2RQ enabled Connection)		
GSN	*6 bits*	*Group Sequence number of first block in the Coded Packet*
Reserved	*5 bits*	*-*
else {		
if (Type bit Extended Type)		*See Table 6*
FSN	*11 bits*	*Sequence number of the current SDU fragment. This field shall increment by one (modulo 2048) for each fragment, including unfragmented SDUs.*
else		
FSN	*3 bits*	*Sequence number of the current SDU fragment.* *This field shall increment by one (modulo 8) for each fragment, including unfragmented SDUs.*
}		
Length	*11 bits*	*Length of the SDU fragment in bytes including packing sub header.*
}		

Insert new subclause 6.3.2.2.7.

6.3.2.2.7 A^2RQ subheader

The A^2RQ subheader (A^2SH) is specified in Table 13m. For the A^2RQ-enabled connection, the A^2RQ subheader is always placed immediately after the FSH and/or PSH.

Table 13m – A^2RQ subheader format

Syntax	Size	Notes
A^2RQ Subheader ( ){		
CPN	*11 bits*	*Sequence number of first block in the CP.*
Coded Packet Size	*10 bits*	*The size of CP (Coded Packet)*

Reserved	*3 bits*	*—*
}		

Renumber clause "6.3.2.2.7" to be "6.3.2.2.8", and renumber the associated subclauses accordingly.

6.3.2.2.8 Extended subheader format

Change the text in Table 13b and 13c as indicated

Table 13b – Description of extended subheader types (DL)

ES type	Name	ES body size	Description
……	……	……	……
5	*PDU SN(long) extended subheader*	*2 bytes*	*See 6.3.2.2.8.8*
6	*A^2RQ extended subheader*	*1 bytes*	*See 6.3.2.2.8.9*
7-127	*Reserved*	*—*	*—*

Table 13c – Description of extended subheader types (UL)

ES type	Name	ES body size	Description
……	……	……	……
5	*Reserved*	*—*	
6	*A^2RQ extended subheader*	*1 bytes*	*See 6.3.2.2.8.9*
7-127	*Reserved*	*—*	*—*

Insert new subclause 6.3.2.2.8.9.

6.3.2.2.8.9 A^2RQ extended subheader

A^2RQ extended subheader specifies the size of CRC (i.e., CRC-16 or CRC-32), and the period of CRC for coded packets (CPs) for the current MAC PDU.

Table 13m - A^2RQ extended subheader

Syntax	Size	Notes
A^2RQ extended sub header ( ){		
CRC Indicator for subheader	*4bit*	*0: absent* *1: CRC 16 present* *2: CRC 32 present* *4: reserve*
CRC period for coded packets	*4bit*	*0: CRC attachment every CP* *1: CRC attachment per 2^1 CPs* *2: CRC attachment per 2^2 CPs* *…..* *15: CRC attachment per 2^{15} CPs*
}		

6.3.2.3 MAC management messages

Change the text in Table 14 as indicated:

Table 14 – MAC Management messages

Type	Message name	Message description	Connection
0			
1			
2			
3			
. . .	. . .	. . .	. . .
66	*MOB_ASC-REP*	*Association result report message*	*Primary management*
67	*A²RQ-Feedback*	*Standalone A²RQ Feedback*	*Basic*
68	*A²RQ-Discard*	*A²RQ Discard message*	*Basic*
69	*A²RQ-Reset*	*A²RQ Reset message*	*Basic*
68-255		*Reserved*	—

Insert new subclause 6.3.2.3.61.

6.3.2.3.61 A²RQ Feedback message

A system supporting A²RQ connection shall be able to receive and process the A²RQ Feedback message. The A²RQ Feedback message can be used to signal any combination of different CIDs. The message shall be sent on the appropriate basic management connection.

Table 109z - A²RQ Feedback message

Syntax	Size	Notes
A²RQ_Feedback_Message_Format(){		
Management Message Type=67	*8 bits*	
A²RQ_Feedback_Payload	*variable*	
}		

A²RQ_Feedback_Payload field shall be either sent using this A²RQ Feedback message or by packing (“Piggybacking”) the A²RQ_Feedback_Payload as described in 6.3.3.4.3.

Insert new subclause 6.3.2.3.62.

6.3.2.3.62 A²RQ Discard message

This message is applicable to A^2RQ-enabled connections only. The transmitter sends this message when it wants to skip a certain number of A^2RQ SDUs (regarding to GSN). The A^2RQ Discard message shall be sent as a MAC management message on the basic management connection of the appropriate direction. Table 109aa shows the format of the Discard message.

Table 109aa – A^2RQ Discard message format

Syntax	**Size**	**Notes**
A^2RQ_Discard_Message_Format () {		
Management Message Type = 68	*8 bits*	
Connection ID	*16 bits*	*CID to which this message refers.*
reserved	*2 bits*	*Shall be set to zero*
GSN	*6 bits*	*Sequence number of the last block in the transmission window that the transmitter wants to discard.*
}		

Insert new subclause 6.3.2.3.63.

6.3.2.3.63 A^2RQ Reset message

This message is applicable to A^2RQ-enabled connection only. The transmitter or receiver may send this message. The message is used in a dialog to reset the parent connection's A^2RQ transmitter and receiver state machines. The A^2RQ Reset message shall be sent as a MAC management message on the basic management connection of the appropriate direction. Table 109ab shows the format of the Reset message.

Table 109ab – A^2RQ Reset message format

Syntax	**Size**	**Notes**
A^2RQ_Reset_Message_Format( ){		
Management Message Type = 69	*8 bits*	
Connection ID	*16 bits*	*CID to which this message refers.*
Type	*2 bits*	*0b00 = Original message from Initiator* *0b01 = Acknowledgement from Responder* *0b10 = Confirmation from Initiator* *0b11 = Reserved*
Direction	*2 bits*	*0b00 = Uplink or downlink*
Reserved	*4 bits*	*Shall be set to zero*
}		

For transport CIDs, the Direction bit shall be set to 0b00 on transmission, and ignored on reception. A^2RQ is not applicable to the secondary management CIDs.

6.3.3 Construction and transmission of MAC PDUs

Insert new subclause 6.3.3.3.3.

6.3.3.3.3 Fragmentation for A²RQ-Based connections

For A²RQ-based connections, fragmentation subheader is used in a way similar to that of ARQ-enabled connections.

The A²RQ subheader (A²SH) is included immediately after FSH and PSH. When the subheader protection is needed, CRC can be inserted behind the last subheader. The presence of the CRC for subheader and the CRC size is determined during the service flow establishment. Figure 15a shows an example of a MAC PDU with A²RQ subheader and CRC protection.

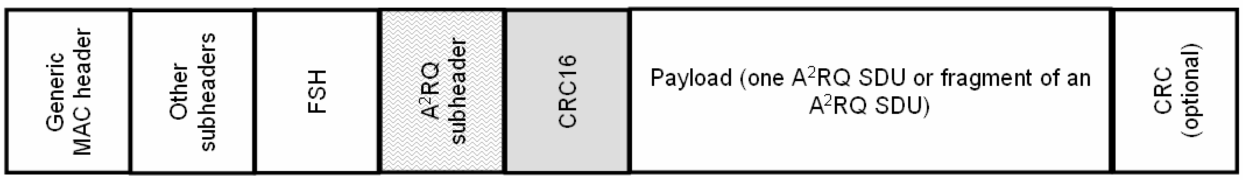

Figure 25a –Example MAC PDU with FSH and A²RQ subheader, and CRC for subheader

Insert new subclause 6.3.3.4.4.

6.3.3.4.4 Packing for A²RQ-enabled connections

For A²RQ-based connections, packing subheader is used in a way similar to that of ARQ-enabled connections. Multiple fragments of multiple A²RQ SDU can be packed into a single MAC PDU, if packing is enabled for a connection.

The A²RQ subheader (A²SH) is included immediately after FSH and PSH. When the subheader protection is needed, CRC can be inserted behind the last subheader. The presence of the CRC for subheader and the CRC size is determined during the service flow establishment.

Figure 30a shows an example of a MAC PDU with A²RQ subheader and CRC protection.

The following figure illustrates the structure of a MAC PDU with A²RQ Packing subheaders. Each of the packed A²RQ SDU or A²RQ SDU fragments or A²RQ feedback payload requires its own Packing subheader, A²RQ subheader and CRC for subheader protection.

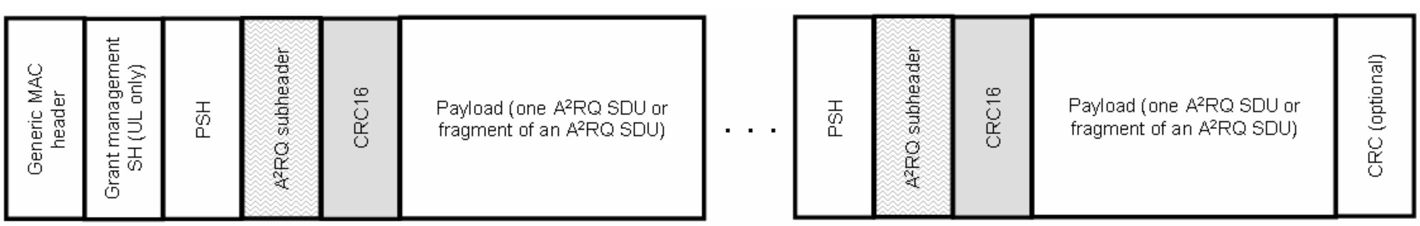

Figure 30a – Example MAC PDU with PSH and A²RQ subheader, and CRC for subheader

6.3.4 ARQ mechanism

Insert following paragraphs at the end of this clause.

A^2RQ can also be used in conjunction with erasure correction code LDPC (ECCL) mechanism to further improve the reliability, link capacity, delay and jitter. Similarly, A^2RQ is an optional feature for implementation. When implemented, it may be enabled on a per-connection basis. The per-connection A^2RQ shall be specified and negotiated during connection creation. A connection cannot have a mixture of A^2RQ with non- A^2RQ traffic. Similar to other properties of the MAC protocol, the scope of a specific instance of A^2RQ is limited to one unidirectional connection.

For A^2RQ-enabled connections, enabling of fragmentation is optional. When fragmentation is enabled, the transmitter may partition each A^2RQ SDU into fragments for separate transmission based on the rule of A^2RQ. When fragmentation is not enabled, the connection shall be managed as if fragmentation was enabled. In this case, each fragment formed for transmission shall contain all the coded packets associated with the parent A^2RQ SDU.

The A^2RQ feedback information can be sent as a standalone MAC management message on the appropriate basic management connection, or piggybacked on an existing connection. A^2RQ feedback cannot be fragmented.

The A^2RQ is a reliable data transmission mechanism using Erasure Correction Code LDPC. A^2RQ creates CPs (Coded Packets) from the A^2RQ SDU, and CPs is consists of SPs (System Packets) and PPs (Parity Packets).

6.3.4.2 ARQ Feedback IE format

Insert new paragraphs and table 111a at the end of this subclause.

A Feedback Payload for A^2RQ consists of one or more A^2RQ Feedback Payload IEs. The Feedback Payload may be sent on an A^2RQ -enabled connection. A set of such IEs of this format may be transported either as a packet payload (“piggybacked”) within a packed MAC PDU or as a payload of a standalone MAC PDU.

Table 111a defines the A^2RQ Feedback IE used by the receiver to signal positive or negative acknowledgement of decoding attempt. In case of the negative acknowledgement, the receiver shall indicate the number of coded packets that have been correctly decoded. Based upon this, the transmitter determines the number of additional parity packets to be transmitted next.

“Received CP Number” IEs are included the A^2RQ Feedback IE to indicate the number of coded packets belonging to the same A^2RQ SDU that can not be decoded correctly at the receiver. The “Received CP Number” shall be organized in the increasing order of GSN value.

Table 111a – A^2RQ Feedback IE

Syntax	Size	Notes
A^2RQ_Feedback_IE(Last){	*Variable*	
CID	*16 bits*	*The ID of the connection being referenced*
LAST	*1 bit*	*0=More A^2RQ feedback IE in the list*

		1=Last A^2RQ feedback IE in the list
Reserved	1 bit	
BGSN	6 bits	Base Group Sequence Number
Number of Active bits	4 bits	0x0=1, 0x1=2, 0x2=3, · · · · 0x0F=16
Reserved	4 bits	
ACK MAP	16 bits	Each bit set to one indicates the corresponding A^2RQ SDU has been decoded successfully. The bit corresponding to the BGSN is MSB of this map.
For (i=0; i<Number of NACK; ++i) {		
Received CP Number	16 bits	Indicate the Received CP number for the A^2RQ SDU with "not succeeded" Group SN.
}		
}		

Number of Active bits
Indicate the number of active bits in the MAP that are used to convey ACK or NACK to the transmitter. The first active bit is the MSB within the MAP entry.

ACK MAP
Each bit that is set to one indicates the corresponding A^2RQ SDU has been decoded successfully. The bit corresponding to the BGSN value in the IE is the most significant bit of the MAP entry. The bits for succeeding GSN are assigned left-to-right (MSB to LSB) within the MAP entry.

6.3.4.3 ARQ parameters

Insert new subclause 6.3.4.3.8.

6.3.4.3.8 A^2RQ_GSN_MODULUS

A^2RQ_GSN_MODULUS is equal to the number of unique GSN value, i.e., 2^6.

Insert new subclause 6.3.4.3.9.

6.3.4.3.9 A^2RQ_CP_SIZE

A^2RQ_CP_SIZE is the value of coded packet (CP). All the system packets (SPs) and parity packets (PPs) created from the same A^2RQ SDU shall have the same length, which is specified by the A^2RQ_CP_SIZE. This parameter is determined on a per A^2RQ SDU basis.

Insert new subclause 6.3.4.3.10.

6.3.4.3.10 A^2RQ_SYNC_LOSS_TIMEOUT

A^2RQ_SYNC_LOSS_TIMEOUT is the maximum time interval A^2RQ_TX_NEXT_GSN/ A^2RQ_RX_HIGHEST_GSN or A^2RQ_TX_NEXT_GSN/ A^2RQ_RX_HIGHEST shall be allowed to remain at the same value before declaring a loss of synchronization of the sender and receiver state machines when data transfer is known to be active. The A^2RQ receiver and transmitter state machines manage independent times. Each has its own criteria for determining when data transfer is "active". Synchronization of the A^2RQ state machine is governed by a timer managed by the transmitter state machine. Each timer A^2RQ GSN is updated, the timer is set to zero. When the timer exceeds the value of A^2RQ_SYNC_LOSS_TIME, the transmitter state machine shall reset the transmitter state variable.

Insert new subclause 6.3.4.3.11.

6.3.4.3.11 A^2RQ_RX_PURGE_TIMEOUT

The A^2RQ_RX_PURGE_TIMER is the timer to detect the time-out to decode. The A^2RQ_RX_PURGE_TIMER is stated when the coded packet advancing GSN is received and this timer is stopped when the decoding of A^2RQ SDU is finished successfully.

Insert new subclause 6.3.4.3.12.

6.3.4.3.12 A^2RQ_WINDOW_SIZE

A^2RQ_WINDOW_SIZE is the maximum number of unacknowledged A^2RQ SDU at any given time. The A^2RQ SDU is considered unacknowledged, if the coded packets created from the A^2RQ SDU have been transmitted but no acknowledgement has been received, given that the acknowledgement is enabled for a A^2RQ connection. A^2RQ_WINDOW_SIZE shall be less than or equal to half of the A^2RQ_GSN_MODULUS.

Insert new subclause 6.3.4.3.13.

6.3.4.3.13 A^2RQ_FEEDBACK_TIMEOUT

A^2RQ_FEEDBACK_TIMEOUT is the protection timer to detect the missing of Feedback Information IEs. A^2RQ_FEEDBACK_TIMEOUT timer is started when the last CP belonging to the same GSN is transmitted, and the timer is stopped when the A^2RQ Feedback IEs are received correctly. When this timer expires, the transmitter detects the state of the missing A^2RQ Feedback IEs and transmits a certain number of parity packets to the receiver.

6.3.4.4 ARQ procedures

Insert new subclause 6.3.4.4.2.

6.3.4.4.2 A^2RQ state machine variables

All A^2RQ state machine variables are set to 0 at connection creation.

Insert new subclause 6.3.4.4.2.1.

6.3.4.4.2.1 Transmitter variables

A^2RQ_TX_WINDOW_START: All A^2RQ SDU with GSN up to (A^2RQ_TX_WINDOW_START - 1) have been acknowledged. By receiving A^2RQ Feedback IEs, the receiver decoded successfully up to (A^2RQ_TX_WINDOW_START -1).

A^2RQ_TX_NEXT_GSN: GSN of the next A^2RQ SDU to encode. This value is incremented one by encoding next A^2RQ SDU. And this value shall reside in the interval A^2RQ_TX_WINDOW_START to (A^2RQ_TX_WINDOW_START + A^2RQ_WINDOW_SIZE), inclusive.

Insert new subclause 6.3.4.4.2.2.

6.3.4.4.2.2 Receiver variables

A^2RQ_RX_WINDOW_START: All A^2RQ SDU with GSN up to (A^2RQ_RX_WINDOW_START -1) have been decoded successfully.

A^2RQ_RX_HIGHEST_GSN: GSN of the highest CP received, plus one. This value shall reside in the interval A^2RQ_RX_WINDOW_START to (A^2RQ_RX_WINDOW_START + A^2RQ_WINDOW_SIZE), inclusive.

6.3.4.5 ARQ-enabled connection setup and negotiation

Insert following paragraph at the end of this subclause.

For A^2RQ, connections are set up and defined dynamically through the DSA/DSC class of messages. The CRC for the subheader protection and the period of CRC for the CPs shall be set. All the A^2RQ parameters shall be set when an A^2RQ-enabled connection is set up. The transmitter and receiver variables shall be reset on connection setup.

6.3.4.6 ARQ operation

6.3.4.6.1 Sequence number comparison

Insert following paragraphs at the end of this subclause.

In the A^2RQ-enabled connection, GSN (Group Sequence Number) and CPN (Coded Packet Number) are used for the sequence number comparison. The CPN is assigned with the value in the range of 0 to 2^{11}-1 for the first CP to the last CP belonging in the same GSN. The GSN is assigned with the value in the range of 0 to 2^6 for the CPs created from the same A^2RQ SDU. The GSN is incremented by 1 for every A^2RQ SDU.

Fragment and Packing is performed by using GSN and CPN. The FSH and PSH contain the GSN of first CP, and A^2RQ SH contains the CPN of first CP.

6.3.4.6.2 Transmitter state machine

Insert following paragraphs at the end of this subclause.

For a A^2RQ connection that acknowledgement is enabled, A^2RQ SDU and CPs created from the A^2RQ SDU may be in one of the following four states – not-sent, outstanding, discarded, and waiting-for-sending parity. Any A^2RQ SDU begins as not-sent. After it sent it becomes outstanding for a periodic time termed A^2RQ_FB_TIMEOUT. While an A^2RQ SDU is in outstanding state, it is either acknowledged and discarded, or transitions to waiting-for-sending-parity after A^2RQ_FB_TIMEOUT or Feedback IEs including decoding failure. An A^2RQ SDU can become waiting-for-sending-parity before A^2RQ_FB_TIMEOUT period expires if it is negatively acknowledged (the decoding fails). An A^2RQ SDU may also be acknowledged or change from waiting-for-sending-parity to discard. All PPs are sent in the side of the transmitter, the A^2RQ SDU transits from waiting-for-sending-parity to discard.

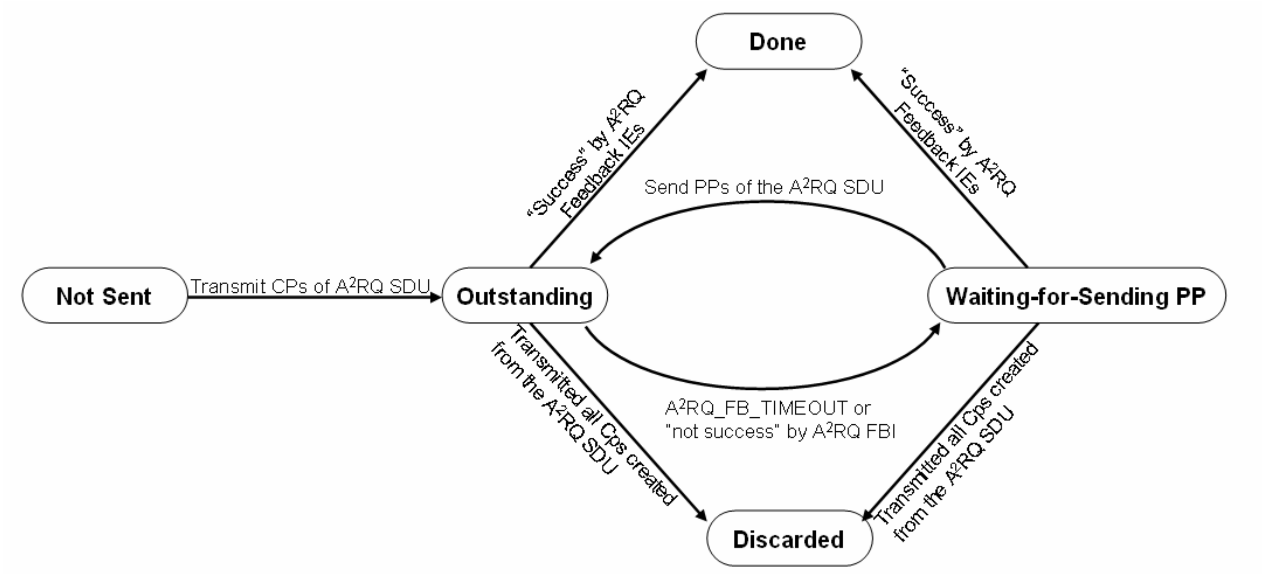

Figure 33a – A^2RQ transmit block states

For an A^2RQ-enabled connection, the transmitter shall first handle (transmit or discard) the PPs of the A^2RQ SDU in "waiting-for-sending-parity" state and only then SPs of the next A^2RQ SDU in the "not-send" state. CPs of the A^2RQ SDU in "outstanding" or "discarded" state shall not be transmitted. When CPs of the A^2RQ SDU are transmitted, the PP with the lowest CPN in the lowest GSN shall be sent first.

A^2RQ SDU and CPs created from the A^2RQ SDU state sequence is shown in Figure 33a.

MAC PDU formation continues with a connection's "not-sent" A^2RQ SDUs. The transmitter builds each MAC PDU using the rules for fragmentation and packing as long as the number of A^2RQ SDUs to be sent plus the number of A^2RQ SDU already transmitted and awaiting transmission of PPs does not exceed the limit imposed by A^2RQ_WINDOW_SIZE. As each "not-sent" A^2RQ SDU is formed and included in a MAC PDU, it is assigned the current value of A^2RQ_TX_NEXT_GSN, which is then incremented.

When the A^2RQ Feedback IEs are received, the transmitter shall check the validity of the GSN. A valid GSN is one in the interval A^2RQ_TX_WINDOW_START to A^2RQ_TX_NEXT_BSN-1 (inclusive). If GSN is not valid, the transmitter shall ignore the acknowledgment.

Synchronization of the A^2RQ state machines is governed by a timer managed by the transmitter state machine. Each time A^2RQ_TX_WINDOW_START is updated, the timer is set to zero. When the timer exceeds the value of A^{2}RQ_SYNC_LOSS_TIMEOUT, the transmitter state machine shall initiate a reset of the connection's state machines as described in Figure 34a.

When in A^{2}RQ reset error state in Figure 34a and Figure 35b, the SS shall re-initialize its MAC, the behavior for BS is implementation dependant.

A Discard message may be sent to the receiver when the transmitter has sent all PPs and wants to skip A^{2}RQ SDU up to the GSN value specified in the Discard message. The message may be sent immediately or may be delayed up to A^{2}RQ_RX_PURTE_TIMEOUT + A^{2}RQ_RETRY_TIMEOUT. Upon receipt of the Discard message, the receiver updates its state information to indicate the specified A^{2}RQ SDU were received and forwards the information to the transmitter through an A^{2}RQ Feedback IE at the appropriate time.

For an A^{2}RQ connection that acknowledgement is disabled, a A^{2}RQ CP may be one of the following 2 states – *not-sent*, and *done*, as shown in Figure 35c. After it is sent, it transits from not sent to done state. When the CP belonging in the new A^{2}RQ SDU is sent, the value of A^{2}RQ_TX_NEXT_GSN is incremented one.

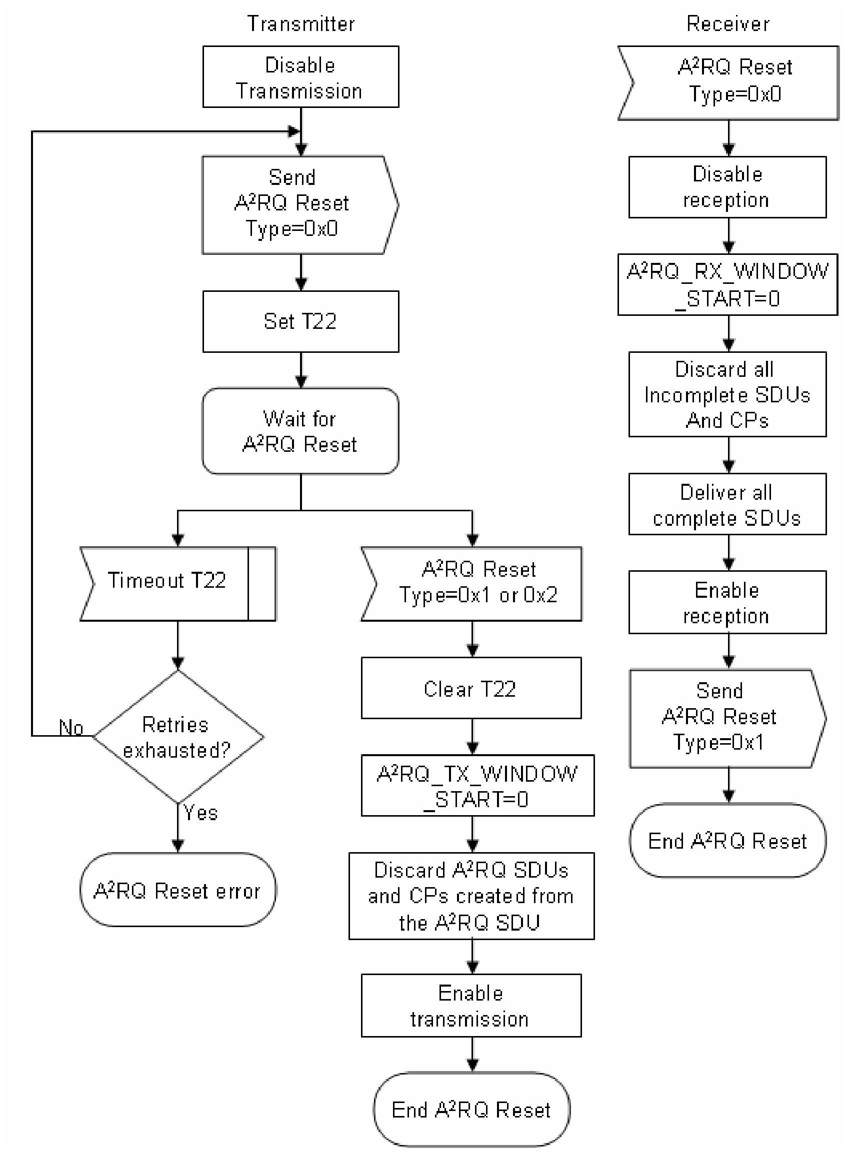

Figure 34a – A^{2}RQ Reset message dialog – initiated by transmitter.

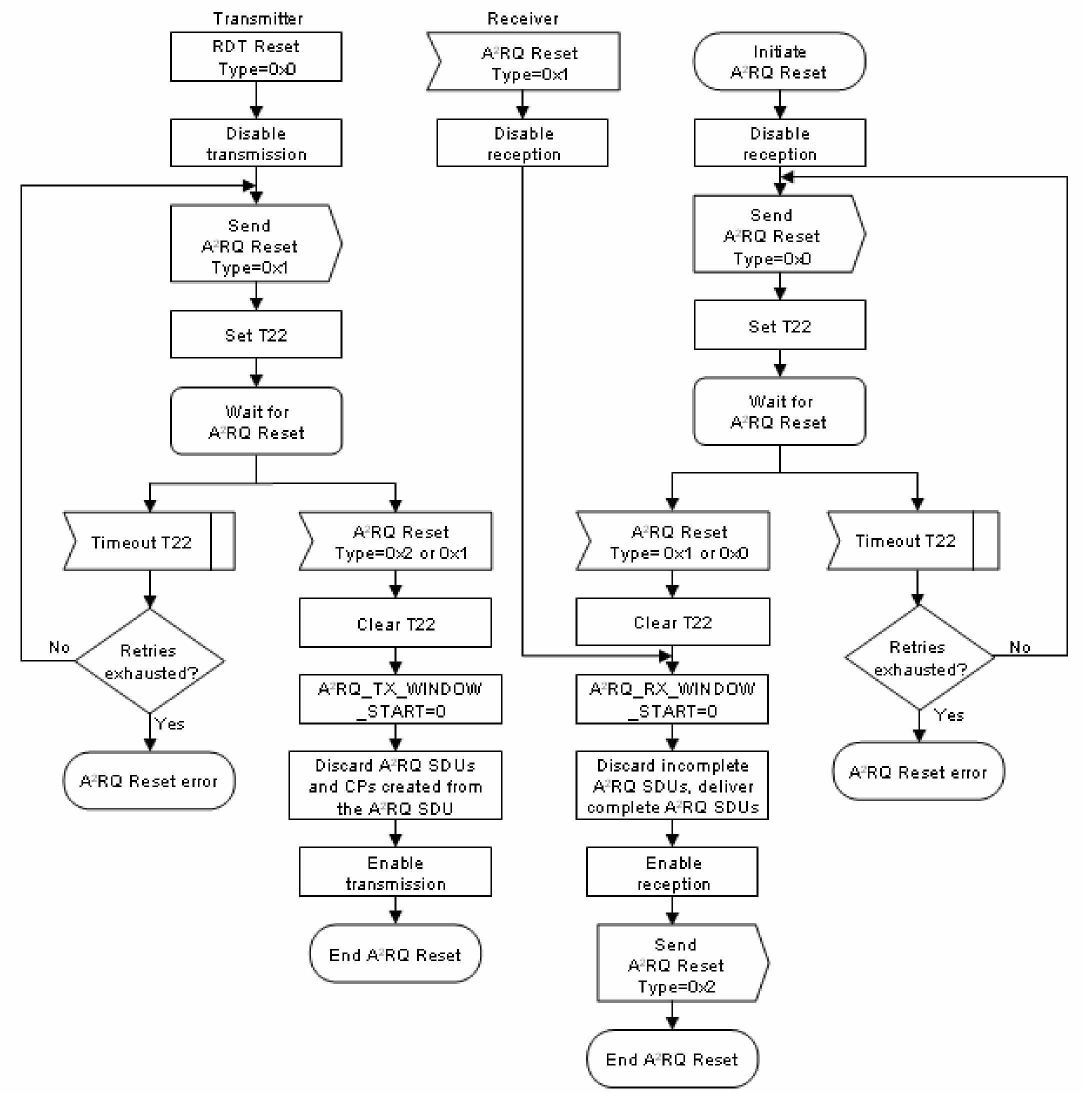

Figure 35b – A^{2}RQ Reset message dialog – initiated by receiver.

Figure 35c – A^{2}RQ CP states.

6.3.4.6.3 Receiver state machine

Insert following paragraphs at the end of this subclause.

For an A^{2}RQ connection that acknowledgement is enabled, the integrity of the received PDU is checked based on the CRC after subheaders and CRC included periodically in the data parts (CPs region). If subheaders and some CPs pass the checksum, it is unpacked and de-fragmented, if necessary. The receiver maintains a sliding-window defined by A^2RQ_RX_WINDOW_START state variable and the A^2RQ_WINDOW_SIZE parameter. When a CP created from the A^2RQ SDU with a number that falls in the range defined by the sliding window is received, the receiver shall accept it. CPs created from the A^2RQ SDU numbers outside the sliding window shall be rejected as out of order.

The sliding window is maintained such that the A^2RQ_RX_WINDOW_START variable always points to the lowest numbered A^2RQ SDU and CPs created from the A^2RQ SDU that has not been decoded successfully. When a A^2RQ SDU with a number corresponding to the A^2RQ_RX_WINDOW_START is decoded successfully, the window is advanced (i.e., A^2RQ_RX_WINDOW_START is incremented modulo A^2RQ_GSN_MODULUS) such that the A^2RQ_RX_WINDOW_START variable points to the next lowest numbered A^2RQ SDU and CPs created from the A^2RQ SDU has not been decoded successfully. The timer associated with A^2RQ_SYNC_LOSS_TIMEOUT shall be reset.

When the CPs don't result in an advance of the A^2RQ_RX_WINDOW_START, the A^2RQ_RX_PURGE_TIMEOUT for the A^2RQ SDU decoded from these CPs shall be started. When the value of the timer for a A^2RQ SDU exceeds A^2RQ_RX_PURGE_TIMEOUT, the timeout condition is marked. When the timeout condition is marked, A^2RQ_RX_WINDOW_START is advanced to the GSN of the next A^2RQ SDU not yet decoded successfully after the marked A^2RQ SDU. Timers for delivered blocks remain active and are monitored for timeout until the GSN values are outside the receiver window.

When the A^2RQ_RX_WINDOW_START is advanced by expire of A^2RQ_RX_PURGE_TIMEOUT, any GSN values corresponding to the A^2RQ SDUs that have not yet been decoded successfully residing in the interval between the pervious and current A^2RQ_RX_WINDOW_START value shall be marked as received and the receiver shall send a A^2RQ Feedback IE to the transmitter with the updated information. And any A^2RQ SDUs decoded successfully in the interval shall be delivered to the upper layer and CPs decoded unsuccessfully in the interval shall be discarded.

When a discard message is received from the transmitter, the receiver shall discard the CPs created from the specified A^2RQ SDU, advance A^2RQ_RX_WINDOW_START to the GSN of the first A^2RQ SDU not yet decoded successfully after the GSN provided in the Discard message, and mark all not decoded correctly in the interval from the previous to new A^2RQ_RX_WINDOW_START values as decoded successfully for A^2RQ Feedback IE reporting.

The result of the decoding to the A^2RQ SDU shall be sent to the transmitter by using GSN. When the receiver can't decode successfully, the number of CPs received correctly with the GSN for the A^2RQ SDU is included in the A^2RQ Feedback IE.

When the A^2RQ SDU is decoded successfully, the A^2RQ SDU shall be segmented and assembled into MAC SDU. MAC SDU shall be transferred to the upper layer. When A^2RQ_DELIVER_IN_ORDER is enabled, MAC SDU is handed to the upper layers from the MAC SDUs with sequence numbers small than MAC PDU segmented and assembled. When A^2RQ_DELIVER_IN_ORDER is not enabled, MAC SDUs are handed to the upper layer in order of segmentation and assemble.

The action to be taken by the receiver state machine when an A^2RQ Reset message is received, are provided in Figure 34a. The action to be taken by the receiver state machine when it wants to initiate a reset of the transmitter A^2RQ state machine, are provided in Figure 35b.

Synchronization of the A²RQ state machines is governed by a timer managed by the receiver state machine. Each timer A²RQ_RX_WINDOW_START is updated, the timer is set to zero. When the timer exceeds the value of A²RQ_SYNC_LOSS_TIMEOUT, the receiver state machine shall initiate a reset of the connection's sate machines as described in Figure 35b.

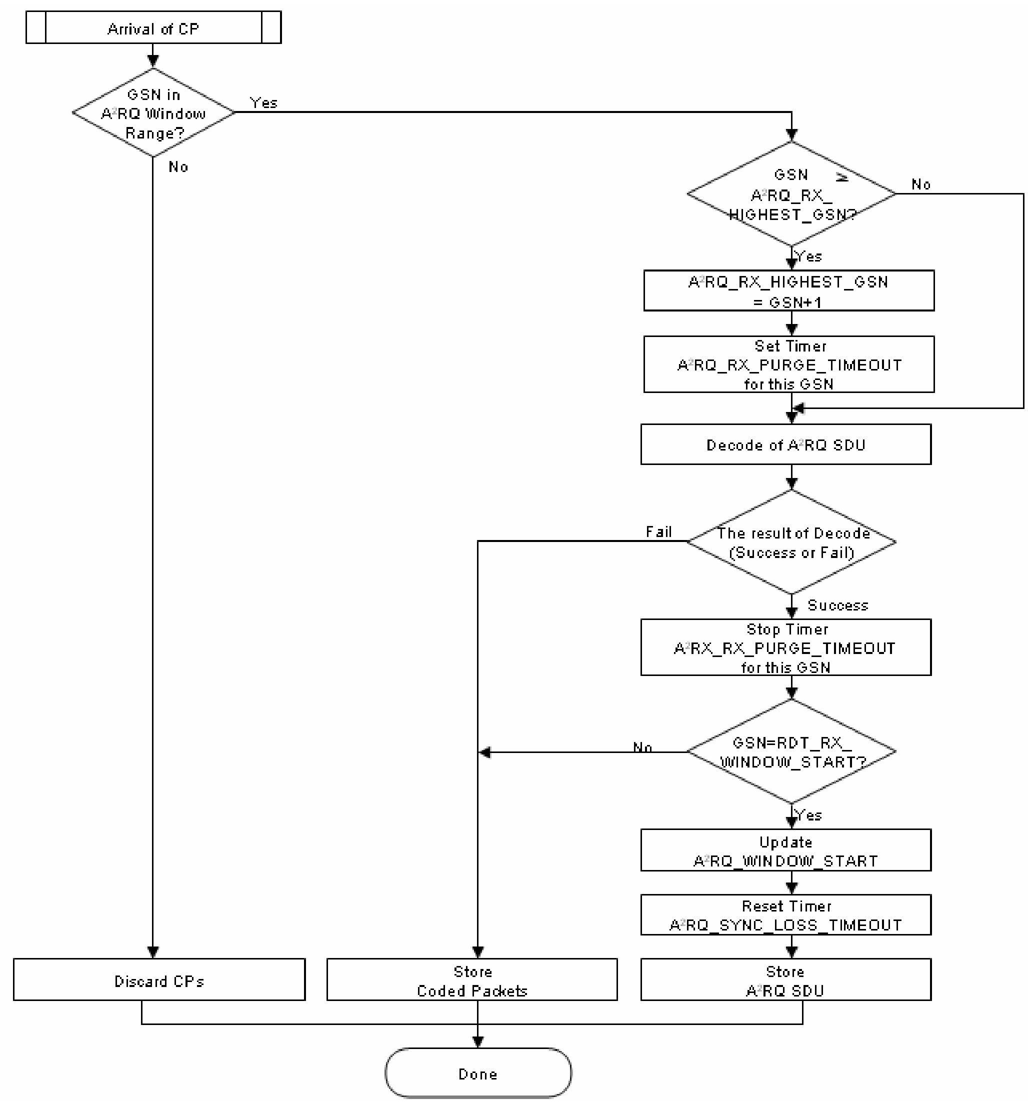

Figure 36b – A²RQ block reception.

For an A²RQ connection that acknowledgement is enabled, when an A²RQ CP is received, its integrity is determined based on CRC. If the CP passes the checksum, the receiver checks the GSN for the CP with A²RQ_RX_HIGHEST_GSN in figure 36b. When the GSN is advanced the A²RQ_RX_HIGHEST_GSN, then A²RQ_RX_HIGHEST_GSN is set to the value of the GSN plus one, and CPs having A²RQ_RX_HIGHEST_GSN-1 is decoded by using ECC LDPC. And A²RQ_RX_PURGE_TIMER is started. And if the FC of FSH or PSH is "last fragment", the receiver shall try to decode as receiving last CPs.

Insert new subclause 6.3.4.6.4.

6.3.4.6.4 Transmitter operation for A²RQ

6.3.4.6.4.1 A^2RQ SDU construction

To form an A^2RQ SDU, multiple protocol data units (i.e., either MSDUs or MPDUs, but not both) may be aggregated at the transmitter, and a new aggregation header is appended in front of each aggregated MSDU or MPDU. The aggregation header is illustrated in Figure 36d and defined in Table 111e.

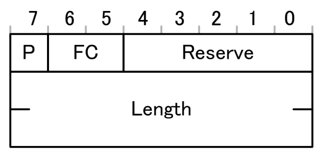

Figure 36d – Format of aggregation header

Table 111e – Format of aggregation header

Syntax	***Size***	***Notes***
PAD Bit	*1 bit*	*The content of next region specified the length IE.* *0: Pad (Next MAC SDU is not concatenated.)* *1: In the next region, the concatenation header of the segment of the next MAC SDU is set*
Segment Status	*2 bits*	*The Segment status of the SDU* *00 : No Segment* *01 : Last Segment* *10 : First Segment* *11 : Middle Segment*
Length	*16 bits*	*The length of segment of SDU (1 – 65535 Byte)*

The maximum size of A^2RQ SDU is limited by the predefined number of system packets (SPN) and predefined maximum size of coded packet (MAX_CODED_PACKET_SIZE). The length of A^2RQ SDU is less than MAX_CODED_SIZE x SPN (= MAX_SDU_LENGTH).

In A^2RQ-enabled connection, the number of system packets (SPN) and the maximum size of coded packet (MAX_CODED_PACKET_SIZE) are specified in the service data flow establishment. The size of coded packet (CP) is from 1 byte to MAX_CODED_PACKET_SIZE, depending on the number of protocol data unit (i.e., MSDU or MPDU) and the size of each unit thereof. In addition, the size of coded packet is fixed in a single A^2RQ SDU, but may vary among different A^2RQ SDU. Padding bits shall be appended at the end of A^2RQ SDU to make its total length a multiple of coded packet number (CPN). An example of aggregation is provided in Figure 36e.

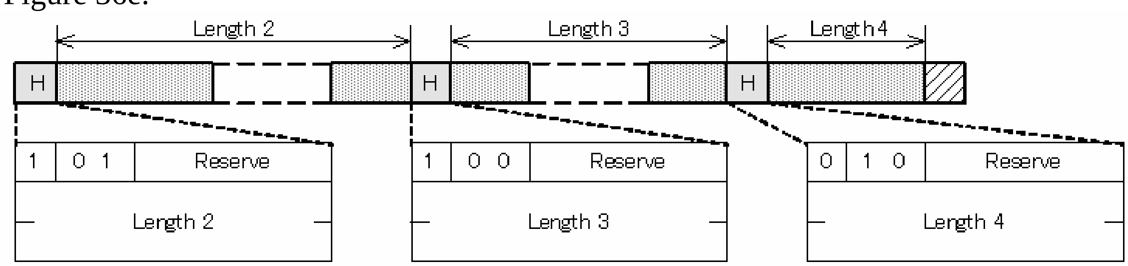

Figure 36e – Example of aggregation for A²RQ

Denote L as the total length of all the protocol data units (i.e., MPDU or MSDU) and the associated aggregation header that will be aggregated to form a single A²RQ SDU. The coded packet size (*CPS*) then can be determined as *ceil[L/CPN]*, where *CPN* (coded packet number) is a system parameter negotiated a prior. Finally, the number of padding bytes needed shall be $(CPS \times CPN) - L$.

6.3.4.6.4.2 Erasure Correction Code encoding

Upon the formation of an A²RQ SDU, it is encoded by using erasure correction code (ECC) LDPC. For example, when the number of system packets is 180, and the ECCL LDPC rate of ½ is used, 180 system packets and 180 corresponding parity packets are created. ECC LDPC supports code rate 1/3 as a minimum code rate, and it is a rate compatible code. The LDPC code is specified in subclause 6.3.4.7.

6.3.4.6.4.3 Sequence number assignment

Each A²RQ SDU is assigned with a distinct group sequence number (GSN), which falls in the range of 0 to 2^6-1. The encoding of an A²RQ SDU shall generate *system packet number* (SPN) of system packets, followed by PPN number of parity packets. All the system packets and parity packets generated for the same A²RQ SDU are of the same length of *CPS* bytes, and each of them are assigned with a distinct *coded packet number* (CPN), which starts from 0 and goes up to 2^{11}-1. For all the coded packets that belong to the same A²RQ SDU, they shall have the same GSN. The GSN and CPN are used together for sequence number comparison. An example of GSN and CPN assignment is shown in Figure 36f.

Fragmentation and packing are performed by using GSN and CPN. The FSH and PSH shall contain the GSN of the first coded packet, while A²RQ SH shall carry the CPN of the first CP. The example of FSH, PSH, and A²RQ SH to assign the GSN and CPN is shown in the Figure 36g.

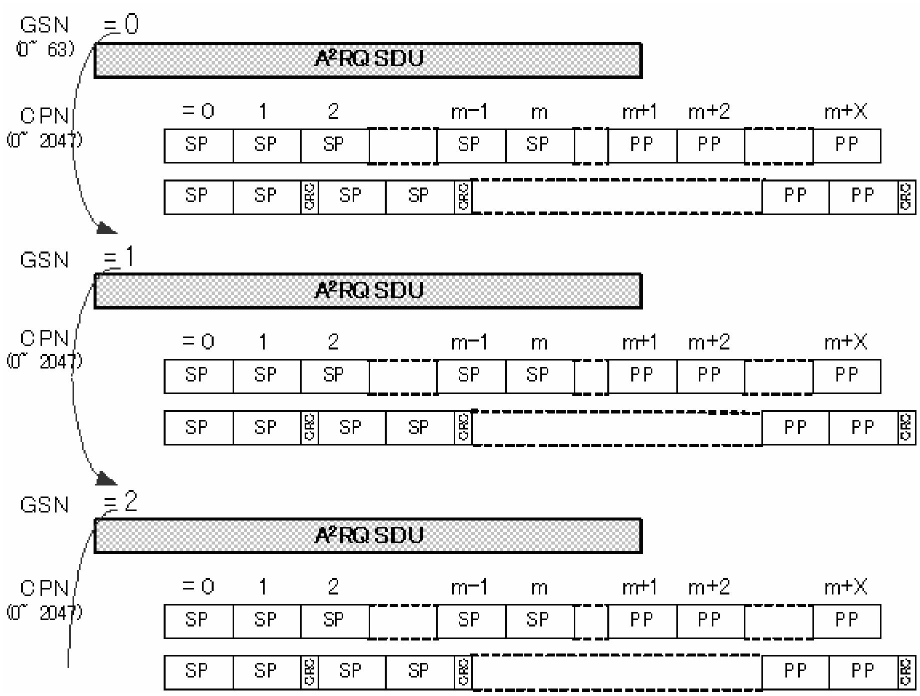

Figure 36f – Example of GSN and CPN assignment

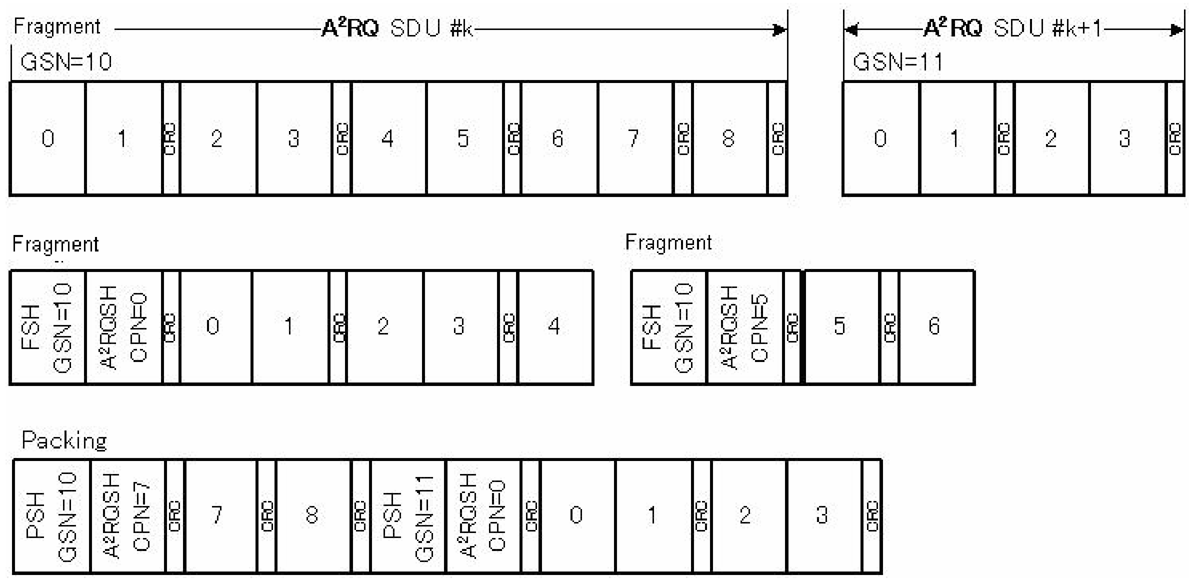

Figure 36g – Example of GSN and CPN in FSH, PSH and RSH.

6.3.4.6.4.4 CRC

The cyclic redundancy check (CRC) that covers every predefined number of coded packets is appended at the end of the last one of the corresponding coded packets. This predefined number is set during the service data flow establishment. In addition, a CRC is always attached at the end of the last coded packets of an A^2RQ SDU. Since the size of coded packets may vary from one A^2RQ SDU to another, the CRC period shall be selected such that the product of *coded packet size* and *CRC period* is no less than a predefined limit. An example of CRC in A^2RQ is shown in the Figure 36h.

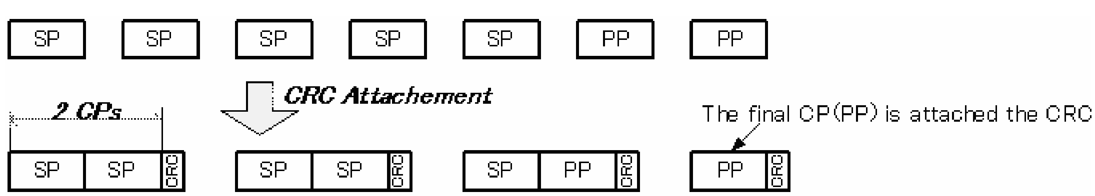

Figure 36h – Example of CRC in A^2RQ.

6.3.4.6.4.5 MPDU construction and transmission

The blocks of coded packets and associated CRC shall be divided to form multiple MPDUs for transmission. Each MPDU created thereof shall contain integer number of coded packets and CRC field(s). In addition, no CRC field can be the first block in a resultant MPDU.

If acknowledgement is enabled in the A^2RQ, all the system packets shall be transmitted first, followed by parity packets, if necessary. Upon the reception of a A^2RQ Feedback IE, the transmitter calculates the number of parity packets to be carried in the next MPDU for transmission. The actual algorithm to determine the number of parity packets to include in the MPDU is implementation dependent. For example, Number of parity packets = MAX [(the number of encoded SPs – the number of CPs received successfully), 0] + (the number of SPs x 0.05), where number 0.05 means the number of the minimum additional PPs in the ECC LDPC.

The transmitter shall stop transmitting parity packets under the following conditions.

1) The Feedback IEs indicates that the target GSN's A^2RQ SDU has been successfully decoded by the receiver.
2) The A^2RQ_RX_PURGE_TIMEOUT timer expires in the side of the receiver, and Feedback IEs include the information decoded successfully for the target GSN's A^2RQ SDU.
3) When transmitter has transmitted all the parity packets.

If acknowledgement is not used in the A^2RQ, all the system packets and parity packets shall be used to construct MPDUs, all of which will be transmitted to the receiver. Thus, a resultant MPDU may contain both system packets and parity packets.

Insert new subclause 6.3.4.6.5.

6.3.4.6.5 Receiver operation for A^2RQ

6.3.4.6.5.1 Receiver operation for A^2RQ

The coded packet size is indicated in the A^2RQ subheader. The CRC period and size used for the received packet shall be negotiated during the service flow establishment. If a CRC period or size different from the ones agreed upon during the service flow setup will be used for an A^2RQ SDU, the new values should be indicated in the extended A^2RQ subheader. Once the CRCs are located, they can be checked to determine whether any coded packets are in error. The correctly received coded packets that belong to a single A^2RQ MSDU at the transmitter side are decoded using ECC LDPC. If the original A^2RQ SDU can be decoded successfully, it will be de-aggregated into constituent protocol unit(s) based on the information contained in the aggregation header.

If acknowledgement is enabled in the A^2RQ, receiver shall transmit A^2RQ Feedback information on a periodic basis, upon the detection of new GSN or upon the reception of the last parity packet of the A^2RQ SDU being currently handled.

If acknowledgement is not enabled in the A^2RQ, receiver shall not return any A^2RQ Feedback information to transmitter.

Insert new subclause 6.3.4.7.

6.3.4.7 Erasure Correction code at MAC layer

A rate compatible LDPC code is used as an erasure correction code in A^2RQ.

Insert new subclause 6.3.4.7.1.

6.3.4.7.1 Code Structure and description of RC-LDPC

The construction of RC-LDPC codes is specified below. The parameter p is a prime number, and the base parity-check matrix over $GF(2)$ with LDGM (Low-Density Generation Matrix) structure is defined by a matrix $\mathbf{H}_B$ of size $M \times N = (pJ)\times(pL + pJ)$ such that

$$\mathbf{H}_B := \begin{bmatrix} I(p_{0,0}) & I(p_{0,1}) & \cdots & I(p_{0,L-1}) & I(0) & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{0} \\ I(p_{1,0}) & I(p_{1,2}) & \cdots & I(p_{1,L-1}) & I(0) & I(0) & \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ I(p_{J/2-1,0}) & I(p_{J/2-1,2}) & \cdots & I(p_{J/2-1,L-1}) & \vdots & \ddots & I(0) & I(0) & \mathbf{0} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & I(0) & \mathbf{0} & \cdots & \mathbf{0} & I(0) & \mathbf{0} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \mathbf{0} & I(0) & \mathbf{0} & \ddots & \ddots & I(0) & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\ I(p_{J-1,0}) & I(p_{J-1,2}) & \cdots & I(p_{J-1,L-1}) & \mathbf{0} & \cdots & \mathbf{0} & I(0) & \mathbf{0} & \cdots & \mathbf{0} & I(0) \end{bmatrix},$$

where for $0 \le j \le J-1,\ 0 \le l \le L-1$, $I(p_{j,l})$ represents the circulant permutation matrix with a one at column-$(r + p_{j,l}) \bmod p,\ (0 \le r \le p-1)$ for row-$r,\ (0 \le r \le p-1)$, and zero elsewhere. It follows that $I(0)$ represents the $p \times p$ identity matrix. And $\mathbf{0}$ is zero matrices of size $p \times p$.

For example, $I(1)$ is as follows,

$$I(1) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Let $\mathbf{H}_{BL}$ be a $M \times (N-M)$ submatrix of left hand side of $\mathbf{H}_B$ such that

$$\mathbf{H}_{BL} := \begin{bmatrix} I(p_{0,0}) & I(p_{0,1}) & \cdots & I(p_{0,L-1}) \\ I(p_{1,0}) & I(p_{1,1}) & \cdots & I(p_{1,L-1}) \\ \vdots & \vdots & \ddots & \vdots \\ I(p_{J-1,0}) & I(p_{J-1,1}) & \cdots & I(p_{J-1,L-1}) \end{bmatrix},$$

where $p_{j,l} = ((p_{0,l} \cdot (j+1)) \bmod p_A) \bmod p$, $p_A = 157$ and for $L = 36$,

For all p,

$p_{0,0} = 146, p_{0,1} = 149, p_{0,2} = 84, p_{0,3} = 141, p_{0,4} = 14, p_{0,5} = 109, p_{0,6} = 14, p_{0,7} = 7, p_{0,8} = 126,$
$p_{0,9} = 119, p_{0,10} = 135, p_{0,11} = 42, p_{0,12} = 22, p_{0,13} = 41, p_{0,14} = 102, p_{0,15} = 102, p_{0,16} = 141, p_{0,17} = 144,$
$p_{0,18} = 145, p_{0,19} = 45, p_{0,20} = 26, p_{0,21} = 124, p_{0,22} = 154, p_{0,23} = 150, p_{0,24} = 42, p_{0,25} = 114, p_{0,26} = 145,$
$p_{0,27} = 155, p_{0,28} = 145, p_{0,29} = 85, p_{0,30} = 156, p_{0,31} = 103, p_{0,32} = 142, p_{0,33} = 131, p_{0,34} = 155, p_{0,35} = 131.$

And let $\mathbf{Z} = [z_{j,l}]$ be a $J \times L$ over $GF(2)$. The product of $\mathbf{Z}$ and $\mathbf{H}_{BL}$ is defined as following:

$$\mathbf{M} = \mathbf{Z} \otimes \mathbf{H}_{BL} = \begin{bmatrix} z_{0,0}I(p_{0,0}) & z_{0,1}I(p_{0,1}) & \cdots & z_{0,L-1}I(p_{0,L-1}) \\ z_{1,0}I(p_{1,0}) & z_{1,1}I(p_{1,1}) & \cdots & z_{1,L-1}I(p_{1,L-1}) \\ \vdots & \vdots & \ddots & \vdots \\ z_{J-1,0}I\left(p_{J-1,0}\right) & z_{J-1,1}I\left(p_{J-1,1}\right) & \cdots & z_{J-1,L-1}I\left(p_{J-1,L-1}\right) \end{bmatrix},$$

where

$$z_{j,l}I(p_{j,l}) = \begin{cases} I(p_{j,l}) & \text{for } z_{j,l} = 1, \\ \mathbf{0} & \text{for } z_{j,l} = 0. \end{cases}$$

This product defines a masking operation for which a set of permutation matrices in $\mathbf{H}_{BL}$ is masked by zero-entries of $\mathbf{Z}$. The distribution of the permutation matrices in $\mathbf{M}$ is the same as the distribution of 1-entries of $\mathbf{Z}$.

The RC-LDPC code $\mathbf{C}$ is defined as the null space of a parity-check matrix $\mathbf{H}_M$ such that:
$\mathbf{H}_M := [\mathbf{M}|\mathbf{H}_T]$,
where

$$\mathbf{H}_T := \begin{bmatrix} I(0) & \mathbf{0} & \cdots & \cdots & \cdots & \cdots & \cdots & \mathbf{0} \\ I(0) & I(0) & \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & I(0) & I(0) & \mathbf{0} & \ddots & \ddots & \vdots \\ I(0) & \mathbf{0} & \cdots & \mathbf{0} & I(0) & \mathbf{0} & \ddots & \vdots \\ \mathbf{0} & I(0) & \mathbf{0} & \ddots & \ddots & I(0) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & I(0) & \mathbf{0} & \cdots & \mathbf{0} & I(0) \end{bmatrix}.$$

Hence, a parity check matrix $\mathbf{M}$ for a LDPC code $\mathbf{C}$ is given by designing only a masking matrix $\mathbf{Z}$.

The information block size $K = N\text{-}M$ and N is the code word block size. Through changing p, a LDPC set of variable information length for various code rates can be obtained.
The parity check matrix of LDPC codes can be fully described by only small parameters of $\boldsymbol{Z}_1^A$ and $p_{0,l}$. $\boldsymbol{Z}_1^A$ is prepared a binary 36×36 matrix for all codeword length.
The masking matrix $\mathbf{Z}$ are designed to be avoided short cycles according to an appropriate degree distribution.

A masking matrix $\mathbf{Z}$ for minimum code rate 1/3 in this specification is shown as follows;

$$\mathbf{Z}_1 = \left[\begin{array}{cc} \multicolumn{2}{c}{\mathbf{Z}_1^A} \\ \hline \mathbf{Z}_1^A\left(1:36,2:13\right) & \mathbf{0}_{36\times24} \end{array} \right],$$

where

$$
Z_1^A=\begin{bmatrix}
1&0&1&0&1&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
1&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&0&0&0\\
1&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
0&1&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0\\
0&1&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1
\end{bmatrix}
$$

$\mathbf{Z}^A(1:36,2:13)$ is a submatrix of $\mathbf{Z}^A$ formed by from 1st row to 36th row and from 2nd column to 13th column, and $\mathbf{0}_{36\times 24}$ is a 36 x 24zero matrix.

$\mathbf{H}_M$ is a parity check matrix $\mathbf{H}_M := [\mathbf{M}|\mathbf{H}_T] = [\mathbf{Z}\otimes\mathbf{H}_{BL}|\mathbf{H}_T]$ for all code word length. The information block sizes of LDPC codes $\mathbf{C}$ is $K = L\times p = 36\times p$, where each p is odd number.

$\mathbf{u}$ and $\mathbf{r}$ are information bits as $\mathbf{u} = (u_1 \quad u_2 \quad \cdots \quad u_K)$ and parity bits as $\mathbf{r} = (r_1 \quad r_2 \quad \cdots \quad r_M)$, respectively. And Let $\mathbf{v}$ be a systematic codeword such that:
$\mathbf{v} := (u_1 \quad u_2 \quad \cdots \quad u_K \quad r_1 \quad r_2 \quad \cdots \quad r_M)$. satisfying $\mathbf{H}_M \cdot \mathbf{v}^T = \mathbf{0}$.

Insert new subclause 6.3.4.7.2.

6.3.4.7.2 Encoding

For $\mathbf{H}_M = [h_{i,j}]$, the syndrome equation $\mathbf{H}_{M_1}\cdot\mathbf{v}^T = \mathbf{0}$ results in the fast-encoding equations below. These three equations show that the parity elements of the codeword are determined from the original sparse parity-check matrix without any need to computing the dense generator matrix.

$$r_l = \begin{cases} \sum_{i=1}^{K} u_i h_{l,i} & 1 \le l \le p \\ r_{l-p} + \sum_{i=1}^{K} u_i h_{l,i} & p < l \le K \\ r_{l-K} + \sum_{i=1}^{K} u_i h_{l,i}, & K < l \le M \end{cases}$$

Insert new subclause 6.3.4.7.3.

6.3.4.7.3 Puncturing

The RC-LDPC encoder consists of a common LDPC encoder and a puncturing device. The decoder for RC-LDPC codes is the same as an ordinary LDPC decoding algorithm with received LLR=0 for puncturing bits.
A set of code rates and un-puncturing bits set $\hat{\boldsymbol{r}}$ for RC-LDPC codes can be represented by:
For the number of un-puncturing bits $i(\le K)$, $\hat{\boldsymbol{r}} = \{\hat{r}_l\}$,

$\boldsymbol{q} = \{q_y : 8,4,6,2,7,3,5,1\}$;

$x = q_0$, $y = 0$;

For $l = 1$ *to* i

If $x \le K$ *then* $\hat{r}_l = r_x$, $x = x + 8$;

Else $y = y + 1$, $x = q_y$;

Endif

End for

For the number of un-puncturing bits $i(> K)$,

$\hat{\boldsymbol{r}} = \{\boldsymbol{r}_l\}$, for $K < l \le i$.

11. TLV encodings

Change the text in the following table as indicated.

Table 383 – Service flow encodings

Type	Parameter
…	………
47	A^2RQ Enable
48	A^2RQ_WINDOW_SIZE
49	A^2RQ_FB_TIMEOUT
50	A^2RQ_SYNC_LOSS
51	A^2RQ_DELIVER_IN_ORDER
52	A^2RQ_PURGE_TIMEOUT
53	A^2RQ_CP_NUMBER
54	A^2RQ_CRC_PERIODIC
55	A^2RQ_CRC_SUBHEADER

56	*MAX_CODED_PACKET_SIZE*
…	………

11.13 Service flow management encodings

Insert new subclause 11.13.38.

11.13.38 A^{2}RQ TLVs for A^{2}RQ-enabled connections

A^{2}RQ technique is not applied to the secondary management connection. So, in the REG-REQ and REG-RSP message sequence, the parameters of the A^{2}RQ are not used.

Insert new subclause 11.13.39.

11.13.39 A^{2}RQ Enable

This TLV indicates whether or not A^{2}RQ use is requested for the connection that is being setup. A value of 0 indicates that A^{2}RQ is not requested and a value of 1 indicates that A^{2}RQ is requested. The DSA-REQ shall contain the request to use A^{2}RQ or not. The DSA-RSP message shall contain the acceptance or rejection of the request. A^{2}RQ shall be enabled for this connection only if both sides report this TLV to be non-zero. The MS shall either reject the connection or accept the connection for the A^{2}RQ.

Type	Length	Value	Scope
[145/146].47	*1*	*0= A^{2}RQ Not Requested/Accepted* *1= A^{2}RQ Requested/Accepted*	*DSA-REQ, DSA-RSP*

Insert new subclause 11.13.40.

11.13.40 A^{2}RQ_Window_Size

This parameter is negotiated upon connection setup or during operation. The DSA-REQ/DSC-REQ message shall contain the suggested value for this parameter. The DSA-RSP/DSC-RSP message shall contain the confirmation value or an alternate value for this parameter. The smaller of the two shall be used as the A^{2}RQ_WINDOW_SIZE.

Type	Length	Value	Scope
[145/146].48	*2*	*>0 and ≤ (A^{2}RQ_BSN_MODULUS/2)*	*DSx-REQ, DSx-RSP*

Insert new subclause 11.13.41.

11.13.41 A^{2}RQ_FB_TIMEOUT

The DSA-REQ message shall contain the value of this parameter as defined by the service flow. If this parameter is set to 0, then the A^{2}RQ_FB_TIMEOUT value shall be considered infinite.

Type	Length	Value	Scope
[145/146].49	*2*	*0=Infinite* *1 —65535 (10 µs granularity)*	*DSA-REQ, DSA-RSP*

Insert new subclause 11.13.42.

11.13.42 A^{2}RQ_SYNC_LOSS_TIMEOUT

The BS shall set this parameter. The DSA-REQ or DSA-RSP message shall contain the value of this parameter as set by the BS. If this parameter is set to 0, then the A^{2}RQ_SYNC_LOSS_TIMEOUT value shall be considered infinite.

Type	Length	Value	Scope
[145/146].50	*2*	*0=Infinite* *1 —65535 (10 µs granularity)*	*DSA-REQ, DSA-RSP*

Insert new subclause 11.13.43.

11.13.43 A^{2}RQ_DELIVER_IN_ORDER

The DSA-REQ message shall contain the value of this parameter. This TLV indicates whether or not data is to be delivered by the receiving MAC to its client application in the order in which the data was handed off to the originating MAC.

Type	Length	Value	Scope
[145/146].51	*1*	*0 —Order of delivery is not preserved* *1 —Order of delivery is preserved*	*DSA-REQ, DSA-RSP*

If this flag is not set, then the order of delivery is not preserved. If this flag is set (to 1), then the order of delivery is preserved.

Insert new subclause 11.13.44.

11.13.44 A^{2}RQ_RX_PURGE_TIMEOUT

The DSA-REQ message shall contain the value of this parameter as defined by the parent service flow. If this parameter is set to 0, then the A^{2}RQ_RX_PURGE_TIMEOUT value shall be considered infinite.

Type	Length	Value	Scope
[145/146].52	*2*	*0=Infinite* *1 —65535 (10 µs granularity)*	*DSA-REQ, DSA-RSP*

Insert new subclause 11.13.44.

11.13.44 A²RQ_CP_NUMBER

This parameter is negotiated upon connection setup or during operation. The DSA-REQ/DSC-REQ message shall contain the suggested value for this parameter. The DSA-RSP/DSC-RSP message shall contain the confirmation value or an alternate value for this parameter. A²RQ_CP_NUMBER is fixed in a certain A²RQ-enabled connection.

Type	Length	Value	Scope
[145/146].53	*2*	*108, 180,…*	*DSA-REQ, DSA-RSP*

Insert new subclause 11.13.45.

11.13.45 A²RQ_CRC_PERIODIC

This parameter is negotiated upon connection setup or during operation. The DSA-REQ/DSC-REQ message shall contain the suggested value for this parameter. The DSA-RSP/DSC-RSP message shall contain the confirmation value or an alternate value for this parameter. A²RQ_CRC_PERIODIC specifies the number of CPs attached a CRC. If this parameter is set to 0, CRCs are not attached for the CPs and MAC PDU has a just only one MAC PDU.

Type	Length	Value	Scope
[145/146].54	*2*	*0=Set CRC for one MAC PDU* *1 —65535*	*DSA-REQ, DSA-RSP*

Insert new subclause 11.13.46.

11.13.46 A²RQ_CRC_SUBHEADER

This parameter is negotiated upon connection setup or during operation. The DSA-REQ/DSC-REQ message shall contain the suggested value for this parameter. The DSA-RSP/DSC-RSP message shall contain the confirmation value or an alternate value for this parameter. If this parameter is set to 0, CRCs for the protection of subheaders are not used, and just only one CRC is attached MAC PDU in the end of MAC PDU. If this parameter is set to 1, CRCs for the protection of subheaders are used.

Type	Length	Value	Scope
[145/146].55	*1*	*0=Set CRC for one MAC PDU* *1=Attach the CRC for the confirmation of Subheaders*	*DSA-REQ, DSA-RSP*

Insert new subclause 11.13.47.

11.13.47 MAX_CODED_PACKET_SIZE

The DSA-REQ message shall contain the value of this parameter as defined by the parent service flow. If this parameter is set to 0, then the MAX_CODED_PACKET_SIZE specifies the predefined default size.

Type	**Length**	**Value**	**Scope**
[145/146].56	*2*	*0=128 (Default)* *1-65535*	*DSA-REQ, DSA-RSP*

5. References

[1] "IEEE Standard for Local and Metropolitan Area Networks − Part 16: Air Interface for Fixed Broadband Wireless Access Systems," IEEE Computer Society and the IEEE Microwave Theory and Techniques Society, October 2004.

[2] "IEEE Standard for Local and Metropolitan Area Networks − Part 16: Air Interface for Fixed Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands," IEEE Computer Society and the IEEE Microwave Theory and Techniques Society, February 2006.

[3] "Harmonized definitions and terminology for 802.16j Mobile Multihop Relay," IEEE 802.16j-06/014r1, October 2006.

[4] 3GPP R1-060910, "Performance improvement of the rate-compatible LDPC codes", March, 2006.

[5] T.Kuze, S.Uchida, K.Sawa, A.Otsuka, F.Ishizu, "A Study of Channel Creation Technology for Cognitive Radio Communication", Proc. Commun. Conf. IEICE, '06, B-17-14, pp524, Sept.2006.

[6] S.Uchida, T.Kuze, A.Otsuka, F.Ishizu, "A Study on Reliable Data Transfer with Erasure Correction Code", Proc. Commun. Conf. IEICE '06, B-17-15, pp525, Sept.2006.