Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Proposal on change in ARQ**
Date Submitted	**2003-09-06**

Source(s)		
	Vladimir Yanover	Alvarion mailto:vladimir.yanover@alvarion.com
	Tal Kaitz	Alvarion tal.kaitz@alvarion.com
	Naftali Chayat	Alvarion naftali.chayat@alvarion.com
	Marianna Goldhammer	Alvarion marianna.goldhammer@alvarion.com
	Lei Wang	Wi-Lan LeiW@Wi-LAN.com
	Ofer Kelman	Airspan okelman@Airspan.com
	Radu Selea	Redline Comm. Radu@redlinecommunications.com
	Robert Nelson	MacPhyModems bob_nelson@ieee.org
	Roger Eline	Intel roger.j.eline@intel.com
	Baraa Al Dabagh	Intel baraa.al.dabagh@intel.com
	Itzik Kitroser	Runcom itzikk@runcom.co.il

Re:	It is a response to IEEE 802.16 Working Group Recirculation Ballot #11b Announcement (IEEE 802.16-03/38)
Abstract	The document contains supporting material for the comment to P802.16d/D3 in the part of ARQ implementation
Purpose	802.16 WG is invited to consider the document in the process of comments resolution
Notice	This document has been prepared to assist IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate text contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.16.
Patent Policy and Procedures	The contributor is familiar with the IEEE 802.16 Patent Policy and Procedures (Version 1.0) <http://ieee802.org/16/ipr/patents/policy.html>, including the statement "IEEE standards may include the known use of patent(s), including patent applications, if there is technical justification in the opinion of the standards-developing committee and provided the IEEE receives assurance from the patent holder that it will license applicants under reasonable terms and conditions for the purpose of implementing the standard." Early disclosure to the Working Group of patent information that might be relevant to the standard is essential to reduce the possibility for delays in the development process and increase the likelihood that the draft publication will be approved for publication. Please notify the Chair <mailto:r.b.marks@ieee.org> as early as possible, in written or electronic form, of any patents (granted or under application) that may cover technology that is under consideration by or has been approved by IEEE 802.16. The Chair will disclose this notification via the IEEE 802.16 web site <http://ieee802.org/16/ipr/patents/notices>.

Proposal on change in ARQ

Vladimir Yanover, Tal Kaitz, Naftali Chayat, Marianna Goldhammer (Alvarion),
Lei Wang (Wi-LAN), Ofer Kelman (Airspan),
Radu Selea (Redline Communications), Robert Nelson (MacPhy Modems),
Roger Eline, Baraa Al Dabagh (Intel), Itzik Kitroser (Runcom)

1. Rationale for changes

After recent changes in 802.16d/e (subchannelization) a new situation arose when Base Station may choose (e.g. because of varying channel conditions or varying load) to allocate just a small number of subchannels to an SS. This may produce a severe restriction to the size of fragments transferred through the channel. For example, for channel width = 3.5 MHz, frame size = 2.5 ms, MAP burst payload = 64 bytes (QPSK ½), TTG = 1 symbol, UL rate = QPSK ½ maximum 48 bytes can be transferred at a time. Taking out Generic MAC Header, Fragmentation Subheader and CRC, we come to max payload size 36 bytes

Existing scheme of fragment-based ARQ does not allow re-fragmentation simply because sequential numbers are assigned to fragments and never change. Nevertheless, when a fragment should be retransmitted, it may happen that BS does not provide allocation of sufficient size to transmit the fragment. There is no mechanism to inform SS why BS decided to provide such allocation, will the situation change in next frame(s) etc. So it is a deadlock that every implementation may resolve differently: discard the fragment each time the situation appears or wait for larger allocation with proprietary timeout value etc. As behavior of this type is not specified in the standard (and it is not simple develop reasonable specification), this effect may negatively impact the system performance.

802.16a standard provides the following recommendation in 6.2.3.3:

“The maximum size of a fragment may be negotiated during or after connection establishment. When a maximum value has been established, the transmitter shall only form fragments whose length is less than or equal to this value even if the pending bandwidth allocation would accept a larger fragment.”

Such a recommendation actually does not solve the problem or the price of the solution is very high. According to above example, middle size packet e.g. 360 bytes should be divided into 10 fragments. As a result, we get an overhead of Fragmentation Subheader (2/36=5%). But the worst thing is necessity to process 10 fragmentation subheaders instead of one thus increasing CPU load 10 times.

It is proposed to return to block-based ARQ (IEEE 802.16a-D1) where block size (not fragment size) may be chosen small enough to fit into MAC frame at the lowest presumed data rate. The block size is known to BS and this information can be used in the allocation policy.

Block-based ARQ allows re-fragmentation so we don’t need anymore fragment packets while allocations are large. This way unnecessary MAC overhead and processing overhead are eliminated.

No change in ARQ state machines needed, so suggested changes in 802.16a-D7 are surprisingly simple: to add a chapter that explains block usage (6.2.4.1 below), replace “ARQ fragment” with “ARQ block” and “FSN” (Fragment Sequence Number) with “BSN” (Block Sequence Number) throughout the text for ARQ-enabled conections.

An important notice: ETSI BRAN has already accepted these changes for HIPERMAN standard. So there is also motive of harmonization between two standards.

In the revision r1 of this document references to 802.16-2001 and 80216a-D7 are replaced with references to P80216-2003_D0.

2. Specific changes in IEEE P802.16d/D3

Editorial instructions are marked by *[red italic]*
Text to be deleted is marked by ~~blue strikethrough~~
Text to be added is marked by red.
All references are done to IEEE P802.16d/D3; in the case a section is absent in IEEE P802.16d/D3, the reference points to P80216-2003_D0.

[Change in Section 3]

~~3.59 ARQ Fragment: A distinct unit of data that is carried on an ARQ-enabled connection. Such a unit is assigned a sequence number, and is managed as a distinct entity by the ARQ state machines. An ARQ fragment may be a complete SDU or may be a portion of an SDU that has been partitioned in accordance with the MAC rules for SDU fragmentation.~~

3.3. ARQ Block: A distinct unit of data that is carried on an ARQ-enabled connection. Such a unit is assigned a sequence number, and is managed as a distinct entity by the ARQ state machines. Block size is a parameter negotiated during connection establishment. ARQ Blocks are formed by logically partitioning SDUs using the agreed upon block size to determine the location of each block boundary. When the length of an SDU is not an integral multiple of the block size, the final block of the SDU holds the bytes remaining between the terminating boundary of the last block formed based on the block size and the end of the SDU.

[Change in Section 6.4.2.1.1, Table 6—Type encodings, line 3]

#3	Extended Type Indicates whether the present Packing or Fragmentation Subheaders, is Extended 1 = Extended, Applicable only to connections where ARQ is enabled 0 = not Extended, Applicable to connections where ARQ is not enabled ~~Indicates whether the present Packing or Fragmentation Subheaders, is Extended~~

[Change in Section 6.4.2.2.1]

Table 8—Fragmentation Subheader format

Syntax	Size	Notes
Fragmentation Subheader() {		
FC	2 bits	Indicates the fragmentation state of the payload: 00 = no fragmentation 01 = last fragment 10 = first fragment 11 = continuing (middle) fragment
If (Type bit Extended Type)		See Table 6
~~FSN~~ BSN	11 bits	Sequence number of the first block in the current SDU fragment. ~~This field increments by one (modulo 8 2048) for each fragment block, including unfragmented SDUs~~. Applicable to connections where ARQ is enabled

Else		
FSN	3 bits	Sequence number of the current SDU fragment This field increments by one (modulo ~~2048~~ 8) for each fragment. Applicable to connections where ARQ is not enabled
Reserved	3 bits	
}		

[Change in Section 6.4.2.2.3]

Table 11—Packing Subheader format

Syntax	Size	Notes
Packing Subheader() {		
FC	2 bits	Indicates the fragmentation state of the payload: 00 = no fragmentation 01 = last fragment 10 = first fragment 11 = continuing (middle) fragment
If (Type bit Extended Type)		See Table 6
~~FSN~~ BSN	11 bits	Sequence number of the first block in the current SDU fragment. This field increments by one (modulo modulo 2048) for each fragment, including unfragmented SDUs. Applicable to connections where ARQ is not enabled
else		
FSN	3 bits	Sequence number of the first block in the current SDU fragment. This field increments by one (modulo 8) for each block Applicable to connections where ARQ is enabled
Length	11 bits	The length in bytes of the MAC SDU or SDU fragment, including the length of the Packing Subheader
}		

[Change in Section 6.4.2.3.31]

This message is applicable to ARQ-enabled connections only.

The transmitter sends this message when it wants to skip a certain number of ARQ ~~fragments~~ blocks. The ARQ Discard message shown in Table 4 shall be sent as a MAC management message on the basic management connection of the appropriate direction. Table 56b shows the format of the Discard message.

Table 59 - ARQ Discard message format

Syntax	Size	Notes
ARQ_Discard_Message_Format() {		
Management Message Type = 34	8 bits	
Connection ID	16 bits	CID to which this message refers
Reserved	5 bits	
~~FSN~~ BSN	11 bits	Sequence Number of the last ~~fragment~~ block in the transmission window that the transmitter wants to discard
}		

[Change in Section 6.4.3.4.2]

The use of packing subheaders for ARQ-enabled connections is similar to that for non-ARQ connections as described in 6.4.3.4.1, except that ARQ-enabled connections shall set the Extended Type bit (see Table 6) in the Generic MAC Header to 1, whereas non-ARQ connections shall set the Extended Type bit to 0. In addition, the

fixed-length packing option (6.4.3.4.1.1) is not supported by ARQ-enabled connections. If packing is turned on for a connection, the MAC may pack multiple MAC SDUs into a single MAC PDU. The transmitting side has full discretion whether or not to pack a group of MAC SDUs and/or fragments in a single MAC PDU. ~~Depending on the ARQ policies, the transmitter may choose to pack multiple fragments of the same SDU into a single MAC PDU, even if there is sufficient bandwidth to send the whole MAC PDU un-fragmented.~~ ~~While this does not change the semantics of the packing, the ARQ protocol may utilize this feature to allow flexibility in retransmission.~~

The packing of variable-length MAC SDUs for the ARQ-enabled connections is similar to that of non-ARQ connections, when fragmentation is enabled. The ~~FSN~~ BSN of the Packing Subheader shall be used by the ARQ protocol to identify and retransmit lost fragments. ~~The primary difference between ARQ and non-ARQ packing is in the interaction with fragmentation.~~

For ARQ-enabled connections, when the type field indicates packing subheaders are in use, fragmentation information for each individual MAC SDU or MAC SDU fragment is contained in the associated Packing Subheader. When the type field indicates that packing is not in use, fragmentation information for the MAC PDU's single payload (MAC SDU or MAC SDU fragment) is contained in the fragmentation header appearing in the message. Figure 31 illustrates the use of Fragmentation subheader without packing.

Generic MAC Header	Other subheaders	Fragmentation subheader	Payload (One SDU or fragment of an SDU)	CRC-32

Figure 31—Example MAC PDU with extended Fragmentation subheaders

Figure 32 illustrates the structure of a MAC PDU with ARQ packing subheaders. Each of the packed MAC SDU or MAC SDU fragments or ARQ feedback payload requires its own packing subheader and some of them may be transmissions while others are re-transmissions.

Generic MAC Header	Grant Management Subheader (UL only)	Packing subheader	Payload (One SDU or SDU fragment or a set of ARQ Feedback IEs))	• • •	Packing subheader	Payload (One SDU or SDU fragment)	CRC-32

Figure 32—Example MAC PDU with ARQ Packing subheader

~~Unlike the non-ARQ case, it is possible to have continuation fragments packed with other fragments in the same MAC PDU for various reasons. For example, in order to support flexible re-transmission, the ARQ mechanism may choose to fragment a MAC SDU into multiple fragments and pack them into the same MAC PDU during the first transmission. Similarly~~ MAC PDU may have fragments from different SDUs, including a mix of first transmissions and retransmissions. The 11-bit ~~FSN~~ BSN and 2-bit FC fields uniquely identify each fragment or nonfragmented SDU.

[Change in Section 6.4.3.4.3]

The FSN / BSN field of the Packing Subheader shall be ignored for the ARQ Feedback Payload and the FC bits shall be set to 00.

[Insert before Section 6.4.4.1 additional heading]

6.4.4.1. ARQ Block Usage

The clause describes the use of blocks for ARQ.

A MAC SDU is logically partitioned into blocks whose length is specified by the connection TLV parameter ARQ_BLOCK_SIZE. When the length of the SDU is not an integer multiple of the connection's block size, the final block of the SDU is formed using the SDU bytes remaining after the final full block has been determined. For purposes of ARQ, handling of partial blocks is identical to that for full blocks with one exception. When a partial block is included in an MPDU, the FC bits of the corresponding fragmentation or packing subheader shall always be set to 01 (last fragment). Once an SDU is partitioned into a set of blocks, that partitioning remains in effect until all blocks of the SDU are successfully delivered to the receiver, or the SDU is discarded by the transmitter state machine.

A set of blocks selected for transmission or retransmission is encapsulated into a PDU. A PDU may contain blocks that are transmitted for the first time as well as retransmitted blocks. If fragmentation is enabled for this connection, the fragmentation shall occur only on ARQ block boundaries. If a PDU is not packed, all the blocks in that PDU must have contiguous block numbers. When a PDU is packed, any sequence of blocks between MAC sub-headers and the sequence of blocks after the last packing sub-header must have contiguous block numbers.

When fragmentation is not enabled, the ARQ block size shall be set to match the fixed SDU size connection setting.

If ARQ is enabled at the connection, Fragmentation and Packing sub-headers contain a BSN, which is the sequence number of the first ARQ block in the sequence of blocks following this sub-header. It is a matter of transmitter's policy whether a set of blocks once transmitted as a single PDU should be retransmitted also as a single PDU or not. Figure NNN illustrates the use of blocks for ARQ transmissions and retransmissions; two options for retransmission presented: with and without rearrangements of blocks.

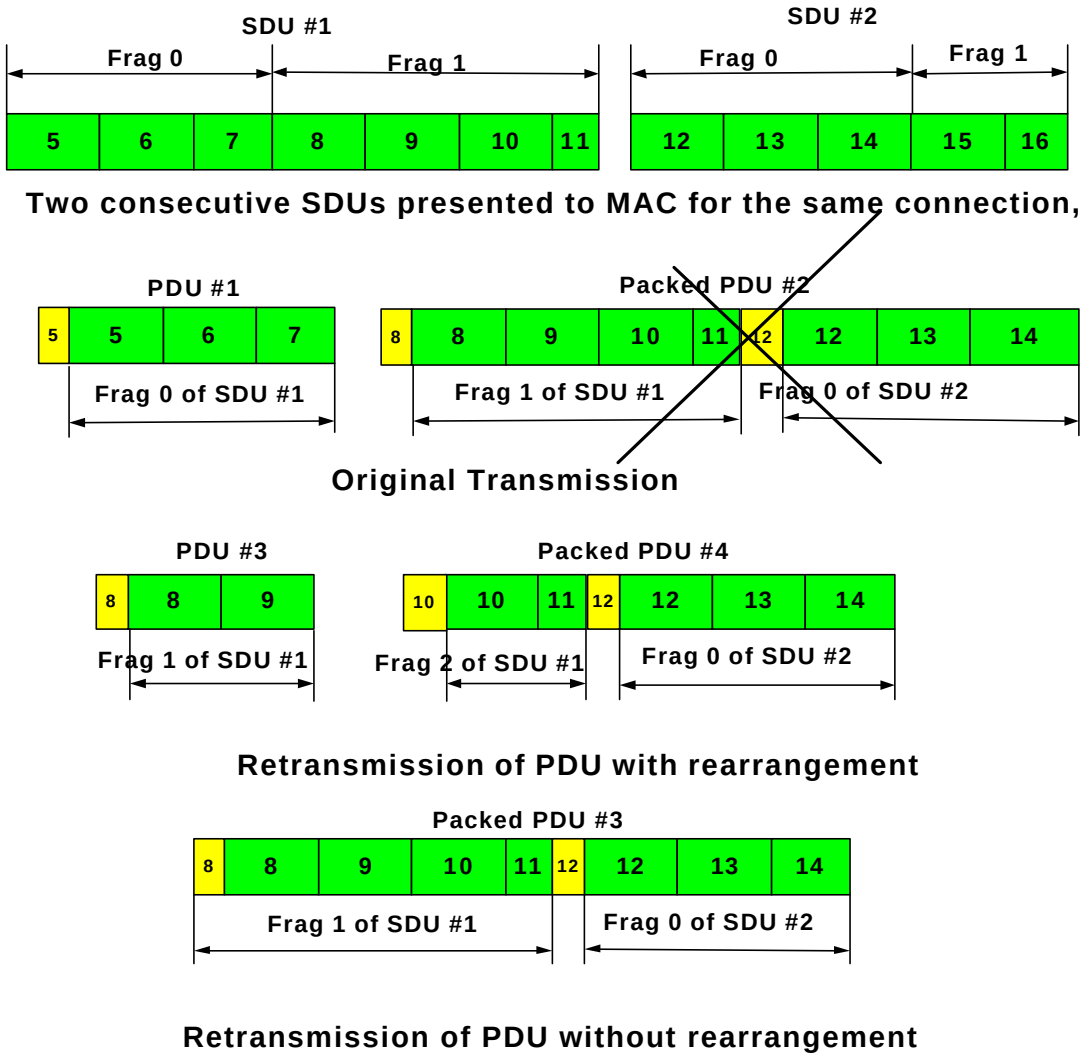

Figure NNN.—Block usage examples for ARQ transmissions and retransmissions

[Change in Section 6.4.4.1]

Table 86 - ARQ Feedback IE

Syntax	Size	Notes
ARQ_feedback_IE (LAST) {	variable	
CID	16 bits	The ID of the connection being referenced
LAST	1 bit	0 = More ARQ feedback IE in the list 1 = Last ARQ feedback IE in the list
ACK Type	2 bit	0x0 = Selective ACK entry 0x1 = Cumulative ACK entry 0x2 = Cumulative with Selective ACK entry 0x3 = Reserved
~~FSN~~BSN	11 bits	
Number of ACK Maps	2 bits	If ACK Type == 01, the field is reserved and set to 00. Otherwise the field indicates the number of ACK maps: 0x0 = 1, 0x1 = 2, 0x2 = 3, 0x3 = 4
if (ACK Type!= 01) {		
for (i=0; i< Number of ACK Maps+ 1; ++i){		
ACK Map	16 bits	
}		
}		
}		

~~FSN~~ BSN
If (ACK Type == 0x0): ~~FSN~~ BSN value corresponds to the most significant bit of the first 16-bit ARQ ACK map.
If (ACK Type == 0x1): ~~FSN~~ BSN value indicates that its corresponding fragment and all fragments with lesser (see 6.2.4.5.1) values within the transmission window have been successfully received.
If (ACK Type == 0x2): Combines the functionality of types 0x0 and 0x1.
ACK Map
Each bit set to one indicates the corresponding ARQ fragment has been received without errors.
The bit corresponding to the ~~FSN~~ BSN value in the IE, is the most significant bit of the first map entry. The bits for succeeding fragment numbers are assigned left-to-right (msb to lsb) within the map entry. If the ACK Type is 0x2, then the most significant bit of the first map entry shall be set to one and the IE shall be interpreted as a cumulative ACK for the ~~FSN~~ BSN value in the IE. The rest of the bitmap shall be interpreted similar to ACK Type 0x0.

[Change in Section 6.4.4.2]

6.4.4.2.1 ARQ_~~FSN~~ BSN _MODULUS

ARQ_~~FSN~~ BSN _MODULUS is equal to the number of unique ~~FSN~~ BSN values, i.e. 2^11.

6.4.4.2.2 ARQ_WINDOW_SIZE

ARQ_WINDOW_SIZE is the maximum number of unacknowledged ARQ ~~fragments~~ blocks at any given time. An ARQ ~~fragment~~ block is unacknowledged if it has been transmitted but no acknowledgement has been received.
ARQ_WINDOW_SIZE shall be less than or equal to half of the ARQ_~~FSN~~ BSN _MODULUS.

6.4.4.2.3 ARQ_FRAGMENT_LIFETIME

ARQ_FRAGMENT_LIFETIME is the maximum time interval an ARQ ~~fragment~~ block shall be managed by the transmitter ARQ state machine, once initial transmission of the fragment has occurred. If transmission (or subsequent retransmission) of the ~~fragment~~ block is not acknowledged by the receiver before the time limit is reached, the ~~fragment~~ block is discarded.

6.4.4.2.4 ARQ_RETRY_TIMEOUT

ARQ_RETRY_TIMEOUT is the time interval a transmitter shall wait before retransmission of an unacknowledged ~~fragment~~ block for retransmission. The interval begins when the ARQ ~~fragment~~ block was last transmitted.

6.4.4.2.5 ARQ_SYNC_LOSS_TIMEOUT

ARQ_SYNC_LOSS_TIMEOUT is the maximum time interval ARQ_TX_WINDOW_START or ARQ_RX_WINDOW_START 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 ARQ receiver and transmitter state machines manage independent timers. Each has its own criteria for determining when data transfer is 'active' (see 6.4.4.5.2 and 6.4.4.5.3).

6.4.4.2.6 ARQ_RX_PURGE_TIMEOUT

ARQ_RX_PURGE_TIMEOUT is the time interval the receiver shall wait after successful reception of a ~~fragment~~ block that does not result in advancement of ARQ_RX_WINDOW_START, before advancing ARQ_RX_WINDOW_START (see 6.4.4.5.3).

6.2.4.2.7. ARQ_BLOCK_SIZE
This value of this parameter specifies the size of ARQ block.

[Change in Section 6.4.4.3]

6.4.4.3 ARQ procedures

6.4.4.3.1 ARQ state machine variables

All ARQ state machine variables are set to 0 at connection creation or by an ARQ reset operation.

6.4.4.3.1.1 Transmitter variables

ARQ_TX_WINDOW_START: All ~~FSN~~ BSN up to (ARQ_TX_WINDOW_START - 1) have been acknowledged.

ARQ_TX_NEXT_~~FSN~~ BSN: ~~FSN~~ BSN of the next ~~fragment~~ block to send. This value shall reside in the interval
ARQ_TX_WINDOW_START to (ARQ_TX_WINDOW_START + ARQ_WINDOW_SIZE), inclusive.

6.4.4.3.1.2 Receiver variables

ARQ_RX_WINDOW_START: All ~~FSN~~ BSN up to (ARQ_RX_WINDOW_START - 1) have been correctly received.
ARQ_RX_HIGHEST_~~FSN~~ BSN: ~~FSN~~ BSN of the highest ~~fragment~~ block received, plus one. This value shall reside in the interval ARQ_RX_WINDOW_START to (ARQ_RX_WINDOW_START + ARQ_WINDOW_SIZE), inclusive.

[Change in Section 6.4.4.5]

6.4.4.5.1 Sequence number comparison

Transmitter and receiver state machine operations include comparing ~~FSNs~~ BSNs and actions taken based on whether it is larger or smaller. In this context, it is not possible to compare the numeric sequence number values directly to make this determination. Instead, the comparison shall be made by normalizing the values relative to the appropriate state machine base value and the maximum value of sequence numbers, ARQ_~~FSN~~BSN_MODULUS, and then comparing the normalized values. Normalization is accomplished by using the expression:

~~fsn' = (fsn - FSN_base) mod ARQ_FSN_MODULUS (8)~~
bsn' = (bsn - BSN_base) mod ARQ_BSN_MODULUS (8)

The base values for the receiver and transmitter state machines are ARQ_TX_WINDOW_START and ARQ_RX_WINDOW_START, respectively.

6.4.4.5.2 Transmitter state machine

~~The transmitter is responsible for choosing the appropriate fragment size on a per-fragment basis. Determining fragment size is outside the scope of this standard. Unlike non-ARQ connections,~~

~~where a single MAC PDU would not normally have two consecutive fragments from the same MAC SDU, this is likely for ARQ enabled connections, since such fragmentation can facilitate retransmission. The MAC SDU fragment structure shall be maintained for retransmissions~~ An ARQ ~~fragment~~ block may be in one of the following four states, not-sent, outstanding, discarded and waiting-for-retransmission. Any ARQ ~~fragment~~ block begins as not-sent. After it is sent it becomes outstanding for a period of time termed ACK_RETRY_TIMEOUT. While a ~~fragment~~ block is in outstanding state, it either is acknowledged and discarded, or transitions to waiting-for-retrans-mission after ACK_RETRY_TIMEOUT or NACK. An ARQ ~~fragment~~ block can become waiting-for-retransmission before the ACK_RETRY_TIMEOUT period expires if it is negatively acknowledged. An ARQ ~~fragment~~ block may also change from waiting-for-retransmission to discarded when an ACK message for it is received or after a timeout ARQ_FRAGMENT_LIFETIME. For a given connection the transmitter shall first handle (transmit or discard) ~~fragment~~ block in 'waiting-for-retransmission' state and only then ~~fragment~~ block in 'non-sent' state. ~~Fragments~~ Blocks in 'outstanding' or 'discarded' state shall not be transmitted. When ~~fragments~~ blocks are retransmitted, the fragment with the lowest ~~FSN~~ BSN shall be retransmitted first. The ARQ transmit ~~fragment~~ block state sequence is shown in Figure 34.

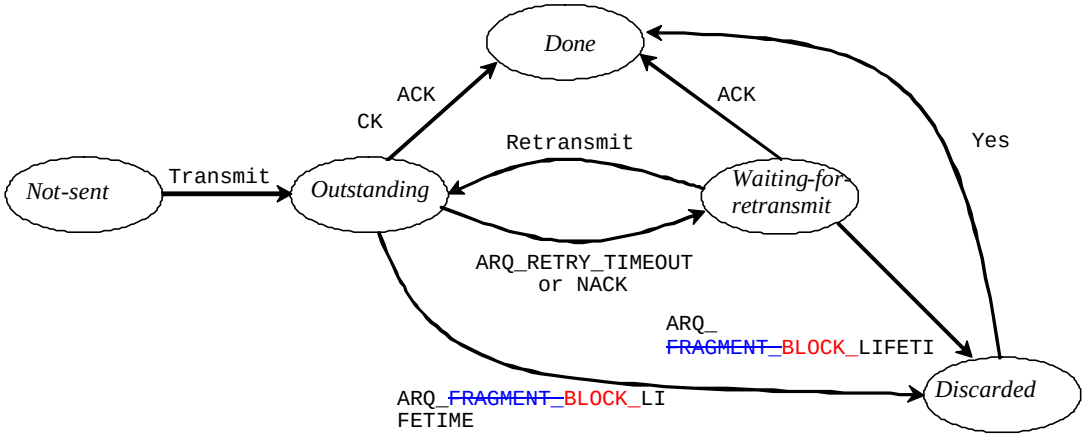

Figure 34: Transmitter ARQ state machine

MAC PDU formation continues with a connection's 'not-sent' MAC SDUs. The transmitter builds each MAC PDU using the rules for fragmentation and packing as long as the number of ~~fragments~~ blocks to be sent plus the number of ~~fragments~~ blocks already transmitted and awaiting retransmission does not exceed the limit imposed by ARQ_WINDOW_SIZE. As each 'not-sent' ~~fragment~~ block is formed and included in a MAC PDU, it is assigned the current value of ARQ_TX_NEXT_~~FSN~~ BSN which is then incremented.
When an acknowledgement is received, the transmitter shall check the validity of the ~~FSN~~ BSN. A valid ~~FSN~~ BSN is one in the interval ARQ_TX_WINDOW_START to ARQ_TX_NEXT_~~FSN~~ BSN -1 (inclusive). If ~~FSN~~ BSN is not valid, the transmitter shall ignore the acknowledgement.

When a cumulative acknowledgement with a valid ~~FSN~~ BSN is received, the transmitter shall consider all ~~fragments~~ blocks in the interval ARQ_TX_WINDOW_START to ~~FSN~~ BSN (inclusive) as acknowledged and set ARQ_TX_WINDOW_START to ~~FSN~~ BSN +1.
When a selective acknowledgement is received, the transmitter shall consider as acknowledged all frag-ments so indicated by the entries in the bitmap for valid ~~FSN~~ BSN values. As the bitmap entries are processed in increasing ~~FSN~~ BSN order, ARQ_TX_WINDOW_START shall be incremented each time the ~~FSN~~ BSN of an acknowledged ~~FSN~~ BSN is equal to the value of ARQ_TX_WINDOW_START.
When ARQ_TX_WINDOW_START has been advanced by either of the above methods and acknowledgement of reception has already been received for the fragment with the ~~FSN~~ BSN value now assigned to ARQ_TX_WINDOW_START, the value of ARQ_TX_WINDOW_START shall be incremented until an ~~FSN~~ BSN value is reached for which no acknowledgement has been received.

A bitmap entry not indicating acknowledged that has an ~~FSN~~ BSN lower than a bitmap entry, which does indicate acknowledged shall be considered a NACK for the corresponding fragment. A not acknowledged bit map entry may also be considered a NACK if sufficient time elapsed before the feedback IE was transmitted to allow the receiver to receive and process the corresponding fragment.

When a cumulative with selective acknowledgement with a ~~FSN~~ BSN is received, the transmitter performs the actions described above for cumulative acknowledgement, followed by those for a selective acknowledgement.

All timers associated with acknowledged fragments shall be cancelled.

A Discard message shall be sent following violation of ARQ_~~FRAGMENT~~BLOCK_LIFETIME. The message may be sent immediately or may be delayed up to ARQ_RX_PURGE_TIMEOUT + ARQ_RETRY_TIMEOUT.
Following the first transmission, subsequent discard orders shall be sent to the receiver at intervals of ARQ_RETRY_TIMEOUT until an acknowledgement to the discarded ~~FSN~~ BSN has been received. Discard orders for adjacent ~~FSN~~ BSN values may be accumulated in a single Discard message.
The actions to be taken by the transmitter state machine when an ARQ Reset Message is received are provided in Figure 35. The actions to be taken by the transmitter state machine when it wants to initiate a reset of the receiver ARQ state machine are provided in Figure 36.

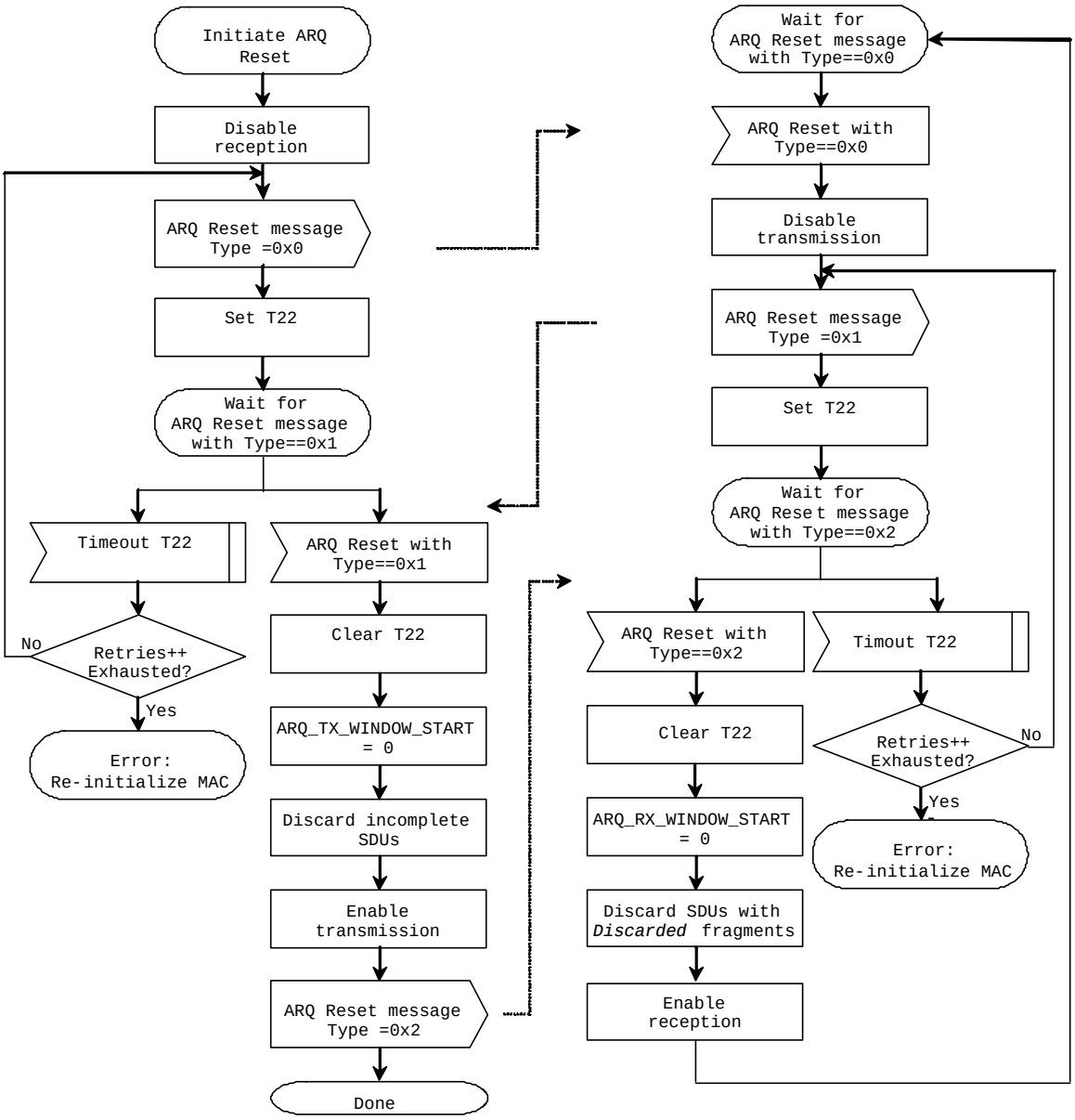

Figure 35—ARQ Reset message dialog - receiver initiated

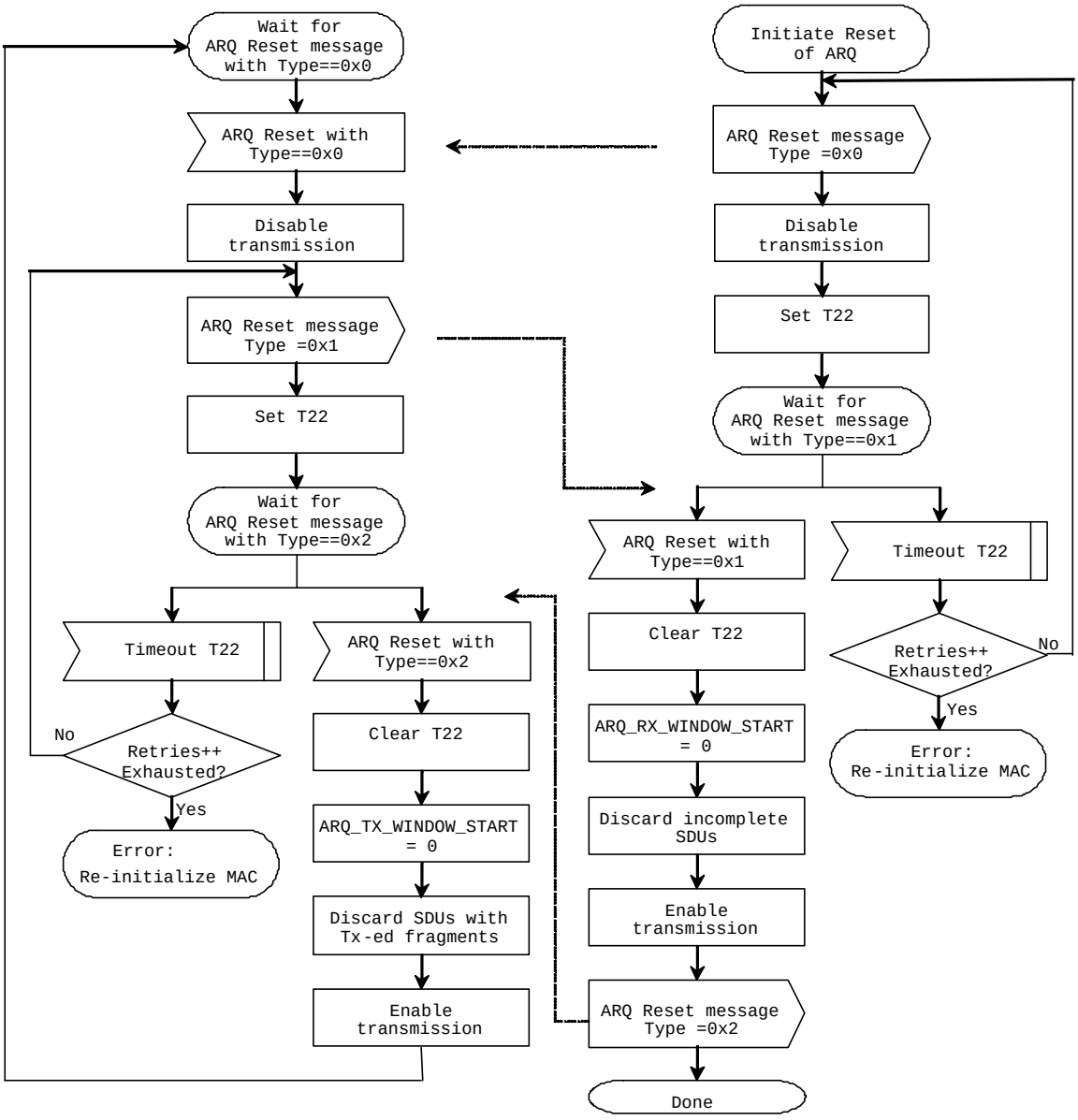

Figure 36—ARQ Reset message dialog - transmitter initiated

Synchronization of the ARQ state machines is governed by a timer managed by the transmitter state machine. Each time ARQ_TX_WINDOW_START is updated, the timer is set to zero. When the timer exceeds the value of ARQ_SYNC_LOSS_TIMEOUT the transmitter state machine shall initiate a reset of the connection's state machines as described in Figure 140.

A Discard message may be sent to the receiver when the transmitter wants to skip ARQ ~~fragments~~ blocks up to the ~~FSN~~ BSN value specified in the Discard message. Upon receipt of the message, the receiver updates its state information to indicate the specified ~~fragments~~ blocks were received and forwards the information to the transmitter through an ARQ Feedback IE at the appropriate time.

6.2.4.5.3 Receiver state machine

When a PDU is received, its integrity is determined based on the CRC-32 checksum. If a PDU passes the checksum, it is unpacked and de-fragmented, if necessary. The receiver maintains a sliding-window defined by ARQ_RX_WINDOW_START state variable and the ARQ_WINDOW_SIZE parameter. When an ARQ ~~fragment~~ block with a number that falls in the range defined by the sliding window is received, the receiver shall accept it. ARQ ~~fragment~~ block numbers outside the sliding window shall be rejected as out of order. The

receiver should discard duplicate ARQ ~~fragments~~ blocks (i.e. ARQ ~~fragments~~ blocks that where already received correctly) within the window.

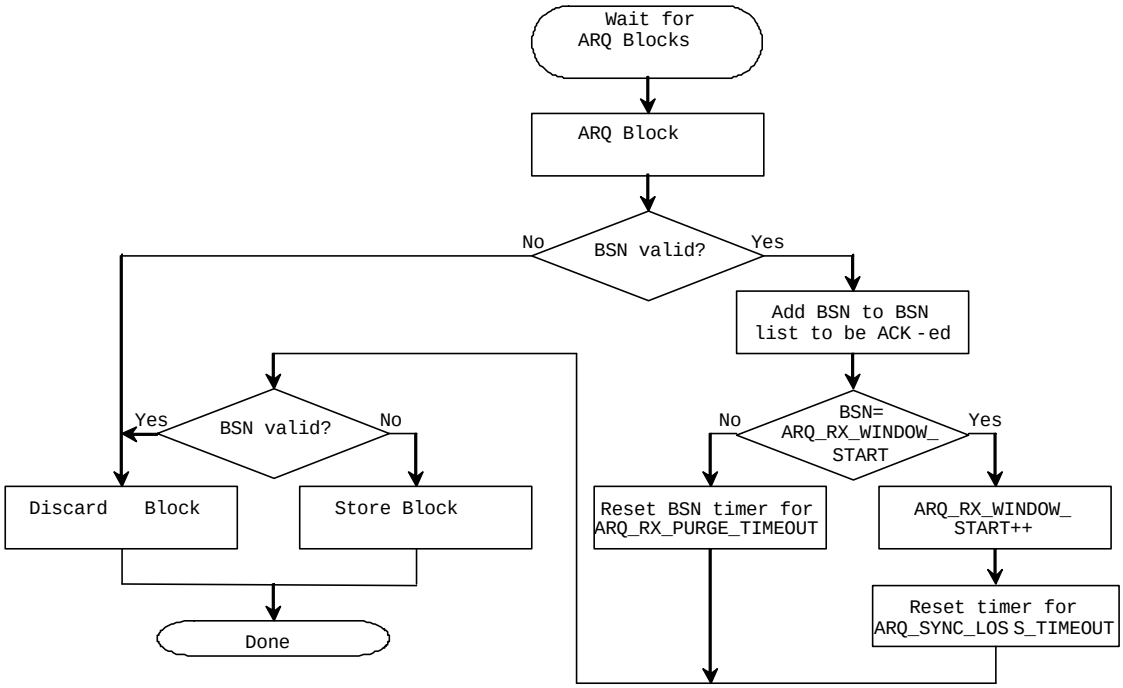

Figure 37 - ARQ fragment block reception

The sliding window is maintained such that the ARQ_RX_WINDOW_START variable always points to the lowest numbered ARQ ~~fragment~~ block that has not been received or has been received with errors. When an ARQ ~~fragment~~ block with a number corresponding to the ARQ_RX_WINDOW_START is received, the window is advanced (i.e. ARQ_RX_WINDOW_START is incremented modulo ARQ_~~FSN~~ BSN _MODULUS) such that the ARQ_RX_WINDOW_START variable points to the next lowest numbered ARQ ~~fragment~~ block that has not been received or has been received with errors. The timer associated with ARQ_SYNC_LOSS_TIMEOUT shall be reset.

As each ~~fragment~~ block is received, a timer is started for that ~~fragment~~ block. When the value of the timer for a ~~fragment~~ block exceeds ARQ_RX_PURGE_TIMEOUT, the timeout condition is marked. When this occurs, ARQ_RX_WINDOW_START is advanced to the FSN of the next ~~fragment~~ block not yet received after the marked ~~fragment~~ block. Timers for delivered ~~fragments~~ blocks remain active and are monitored for timeout until the FSN values are outside the receive window.

When ARQ_RX_WINDOW_START is advanced, any ~~FSN~~ BSN values corresponding to ~~fragments~~ blocks that have not yet been received residing in the interval between the previous and current ARQ_RX_WINDOW_START value shall be marked as received and the receiver shall send an ARQ Feedback IE to the transmitter with the updated information. Any ~~fragments~~ blocks belonging to complete SDUs shall be delivered. ~~Fragments~~ blocks from partial SDUs shall be discarded.

When a discard message is received from the transmitter, the receiver shall discard the specified ~~fragments~~ blocks, advance ARQ_RX_WINDOW_START to the FSN of the first ~~fragment~~ block not yet received after the FSN pro-vided in the Discard message, and mark all not received ~~fragments~~ blocks in the interval from the previous to new ARQ_RX_WINDOW_START values as received for ARQ feedback IE reporting.

For each ARQ ~~fragment~~ block received, an acknowledgment shall be sent to the transmitter. Acknowledgment for ~~fragments~~ blocks outside the sliding window shall be cumulative. Acknowledgments for ~~fragments~~ blocks within the sliding window may be either for specific ARQ ~~fragments~~ blocks (i.e. contain information on the acknowledged ARQ ~~fragment~~ block numbers), or cumulative (i.e. contain the highest ARQ ~~fragment~~ block number below which all ARQ ~~fragments~~ blocks have been received correctly) or a combination of both (i.e., cumulative with selective). Acknowledgments shall be sent in the order of the ARQ ~~fragment~~ block

numbers they acknowledge. The frequency of acknowledgement generation is not specified here and is implementation dependent.
A MAC SDU is ready to be handed to the upper layers when all of the ARQ ~~fragments~~ blocks of the MAC SDU have been correctly received within the time-out values defined.

When ARQ_DELIVER_IN_ORDER is enabled, a MAC SDU is handed to the upper layers as soon as all the ARQ ~~fragments~~ blocks of the MAC SDU have been correctly received within the defined time-out values and all ~~fragments~~ blocks with sequence numbers smaller than those of the completed message have either been discarded due to time-out violation or delivered to the upper layers.

When ARQ_DELIVER_IN_ORDER is not enabled, MAC SDUs are handed to the upper layers as soon as all ~~fragments~~ blocks of the MAC SDU have been successfully received within the defined time-out values.
The actions to be taken by the receiver state machine when an ARQ Reset Message is received are provided in Figure 36. The actions to be taken by the receiver state machine when it wants to initiate a reset of the transmitter ARQ state machine are provided in Figure 35.

Synchronization of the ARQ state machines is governed by a timer managed by the receiver state machine.

Each time ARQ_RX_WINDOW_START is updated, the timer is set to zero. When the timer exceeds the value of ARQ_SYNC_LOSS_TIMEOUT the receiver state machine shall initiate a reset of the connection's state machines as described in Figure 35.

[Add at the end of 6.4.3.3]

For non-ARQ connections, fragments are transmitted once and in sequence. The sequence number assigned to each fragment allows the receiver to recreate the original payload and to detect the loss of any intermediate fragments. Upon loss, the receiver shall discard all MAC PDUs on the connection until a new first fragment is detected or a non-fragmented MAC PDU is detected.

For ARQ-enabled connections, fragments are formed for each transmission by concatenating sets of ARQ blocks (see section 6.4.4.1) with adjacent sequence numbers. The sequence number for the fragment is assigned by taking the block sequence number (BSN) of the first ARQ block appearing in the fragment

[Add new heading]

11.4.9.18.8 ARQ_BLOCK_SIZE

This value of this parameter specifies the size of ARQ block. This parameter is established by negotiation during the connection creation and connection change dialogs.
The requester includes its desired setting in the REQ message. The receiver of the REQ message shall take the smaller of the value it prefers and value in the REQ message. This minimum value is included in the RSP message and becomes the agreed upon length value.
Absence of the parameter during a DSA dialog shall indicate the originator of the message desires the maxi-mum value. Absence of the parameter during a DSC dialog indicates the current setting shall remain in force.

Type	Length	Value	Scope
[24/25].22	2	0- Reserved 1-2040 Desired/Agreed size in bytes 2041-65535 Reserved	DSA-REQ DSA-RSP DSC-REQ DSC-RSP