Project	**IEEE 802.16 Broadband Wireless Access Working Group** <http://ieee802.org/16>	
Title	**Rate Matching in 802.16m**	
Date Submitted	**2008-07-07**	
Source(s)	Keith Blankenship, Mark Cudak, Fred Vook, and Fan Wang Motorola	Voice: E-mail: Keith.Blankenship@motorola.com *<http://standards.ieee.org/faqs/affiliationFAQ.html>
Re:	TGm Call for comments on SDD, IEEE 802.16m-07/047, in the area of "Hybrid ARQ (PHY aspects)"	
Abstract	The convolutional turbo code and subpacket generation blocks currently incorporated in 802.16e are used to generate a high performance error correction code for a small set of data block sizes and code rates. IEEE 802.16m frame structures will require added flexibility in order to achieve the higher spectral efficiency targets. The flexibility and performance of this error correction code can be enhanced by eliminating the 802.16e requirement to support a small set of fixed code rates and expanding the set of allowed data block sizes.	
Purpose	review and adopt	
Notice	*This document does not represent the agreed views of the IEEE 802.16 Working Group or any of its subgroups*. It represents only the views of the participants listed in the "Source(s)" field above. It is offered as a basis for discussion. It is not binding on the contributor(s), who 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 material 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	The contributor is familiar with the IEEE-SA Patent Policy and Procedures: <http://standards.ieee.org/guides/bylaws/sect6-7.html#6> and <http://standards.ieee.org/guides/opman/sect6.html#6.3>. Further information is located at <http://standards.ieee.org/board/pat/pat-material.html> and <http://standards.ieee.org/board/pat>.	

Rate Matching in 802.16m

Keith Blankenship, Mark Cudak, and Fred Vook
Motorola

1. Overview

Figure 1 illustrates the fundamental operations in the channel coding of a CTC-encoded PDU in 802.16e. The PDU is first segmented into a number of data blocks. The size of a data block is generically referred to as *D* in this contribution. Each data block is independently encoded by a convolutional turbo code (CTC) encoder to produce 3*D* encoded bits, which is followed by a subpacket generation procedure that selects a size-*E* subset of the encoded bits to transmit over the channel. The bits of the encoded data block are used to select complex modulation symbols and are subsequently mapped to physical channel resources.

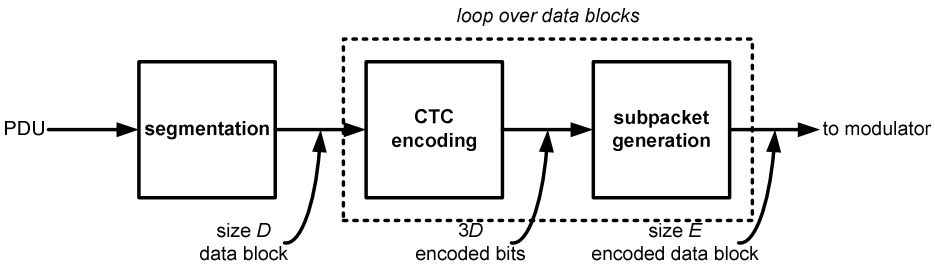

Figure 1. Fundamental operations in the channel coding of a PDU.

802.16e defines two CTC encoding modes, Chase and IR (incremental redundancy), which specify the allowed set of data block sizes and the behavior of the subpacket generation procedure. In particular, the Chase mode of 802.16e defines a set of 12 allowed CTC-encoded data block sizes (between 6 to 60 bytes, inclusive), 4 allowed code rates (1/2, 2/3, 3/4, and 5/6), and 3 allowed modulation orders (QPSK, 16-QAM, and 64-QAM). However, these three parameters are not allowed to vary independently. A few examples of this are as follows:

- a code rate of rate-2/3 is not allowed with 16-QAM modulation.
- a 60-byte data block is not allowed with 16-QAM modulation at a code rate of 3/4.

Without going into details, the IR CTC encoding mode also imposes limitations on the independence of the data block size, code rate, and modulation order.

The inflexibility imposed by supporting a small set of exact code rates, or more generally by limiting the data block size/code rate/modulation order independence, creates inefficiencies in the allocation of time-frequency resources. IEEE 802.16m frame structures must be highly efficient given the spectral efficiency targets in the SRD and, unlike 802.16e, the number of data-bearing subcarriers per allocable element will not be constant. For instance, Motorola is proposing allocable elements called “resource tiles” that contain 108 subcarriers and comprise data, pilot, and control. In Motorola’s proposal, the number of pilots per resource tile may depend on the specific spatial mode employed and velocity. In addition, control channels and uplink feedback may be multiplexed within a resource tile. Other factors such a guard intervals or legacy field support (e.g., preamble) may affect the number of data-bearing subcarriers per resource tile as well.

2. Eliminating the Exact Code Rate Requirement

To solve the problem of channel coding for allocable elements with non-constant data capacity, it is recommended that *802.16m eliminate the requirement to support a set of exact code rates*. Instead, in 802.16m

data block size *D* and encoded block size *E* (the ratio of which is code rate) should be treated as independent parameters. To operate in this manner, the functionality already present in the 802.16e subpacket generation procedure should be exploited.

Fi gur e 2 illustrates the 802.16e subpacket generation procedure. For a data block containing *D* bits, the convolutional turbo code (CTC) naturally generates 6 size-*D*/2 encoded bit streams labeled A, B, Y_1, Y_2, W_1, and W_2. After interleaving and interlacing, a subset of the bits is selected for transmission over the channel. For instance, the figure shows that on the first transmission, a subset of size E1 is selected for transmission over the channel. The size of the subset is chosen to exactly match the coded bit capacity of the physical channel allocation for the data block. For example, if a data block is allotted 100 subcarriers and 16-QAM modulation is used, then 400 encoded bits are selected. In particular, no requirement need be imposed that the resulting code rate, *D*/*E*, be a member of a predefined set of allowable code rates.

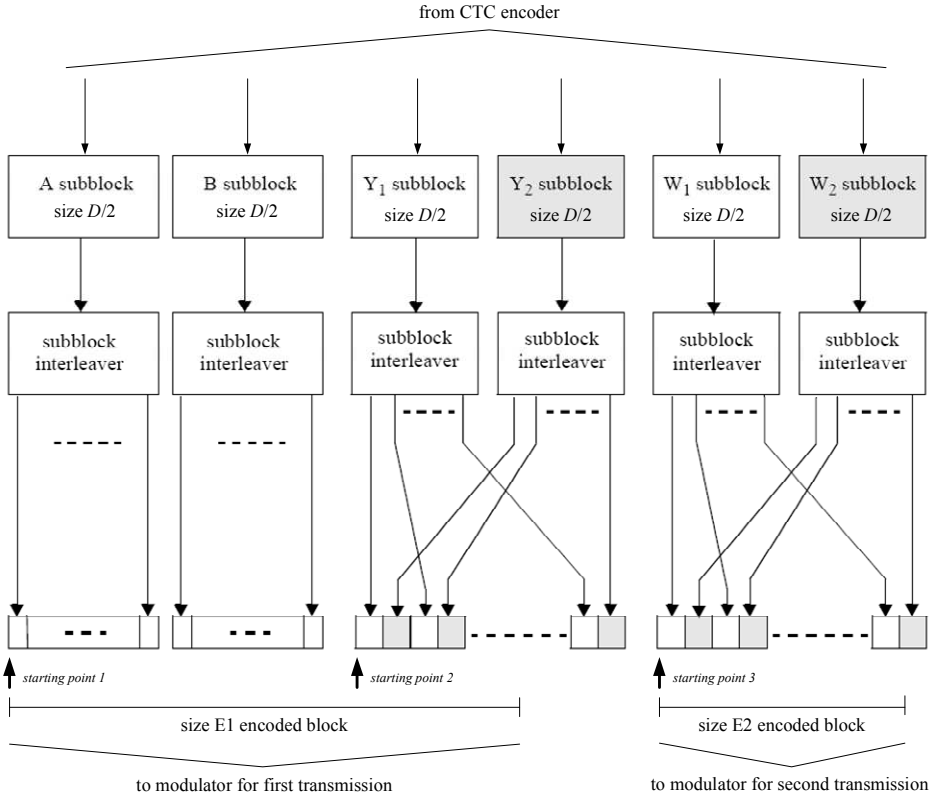

Figure 2. Subpacket generation procedure.

The subpacket generation procedure provides a means of achieving good channel coding performance at arbitrary code rates. In addition, by defining a number of "starting points" on the interleaved/interlaced bits, IR hybrid ARQ (HARQ) can be supported as a natural extension of the subpacket generation procedure. For example, Fi gur e 2 illustrates a case where three starting points are defined on the interleaved/interlaced bits. For the first transmission, a size-E1 set of encoded bits is selected beginning from starting point 1. For the second transmission, a size-E2 set of encoded bits is selected beginning from starting point 3.

3. Expanding the Set of Data Block Sizes

The Chase mode of 802.16e defines a maximum data block size of 60 bytes. Recalling that Internet packets typically have lengths of several hundred to over a thousand bytes, the impact of the 60 byte maximum is that

that in 802.16e Internet packets must typically be segmented into many data blocks for transmission across the channel. For illustrative purposes, consider a scenario where each data block has an independent failure probability of p_{blk}. Then a packet that is segmented into F data blocks as a failure probability, p_{pkt}, of

$$p_{pkt} = 1-(1-p_{blk})^F$$

Note that p_{pkt} is approximately proportional to F for small p_{blk}. Therefore, segmenting packets into a large number of data blocks is detrimental to performance.

Another shortcoming of the 60-byte (480-bit) maximum is that it does not exploit the inherent property of turbo codes of improving coding gain with increasing block size. 60 bytes is generally considered to be at the small end of the spectrum of block sizes over which a turbo code is effective. Instead, turbo codes are effective for block sizes on the order a several thousands of bits.

The HARQ IR mode of 802.16e defines a set of 12 data block sizes between 6 and 600 bytes. In general, increasing the maximum data block size to 600 bytes improves performance. However, due to the coarse granularity of the set, PDUs often do not closely match one of the available sizes. In such cases, up to 120 non-information padding bytes are appended (and subsequently transmitted) in order to make up the difference between the PDU size and the next largest allowed block size. The impact of padding inefficiency could be mitigated by decreasing the granularity of the set of allowed data block sizes.

Therefore, it is recommended that *802.16m expand the set of allowed data block sizes* by extending the maximum data block size up to 768 bytes (approximately half of a 1500-byte TCP/IP packet) and by providing 100-200 allowed data block sizes between 6 and 768 bytes. This should provide the following:

- large data block sizes to minimize segmentation penalties.
- large data block sizes to exploit turbo coding gains.
- a set of data block sizes with sufficiently fine granularity to minimize padding inefficiencies.

4. Conclusions

The present requirement in 802.16e to support a small set of exact code rates will not work in 802.16m where the number of data-bearing subcarriers per allocable element may not be constant. To solve this problem, this contribution has recommended that 802.16m eliminate the exact code rate requirement, instead relying on the functionality already present in the 802.16e subpacket generation procedure to select a number of encoded bits that exactly matches the encoded bit capacity of the allocated resources. The subpacket generation procedure also naturally supports IR HARQ, a well known technique for improving link and system throughput over Chase HARQ.

For 802.16e Chase mode, because the maximum block size is relatively small, the current set of allowed data block sizes imposes significant segmentation penalties and does not exploit the benefits of turbo codes over large block sizes. Although the 802.16e IR mode includes larger block sizes, the coarse granularity of the allowed block sizes introduces padding inefficiencies. To improve performance, this contribution has recommended that 802.16m expand the set of allowed data block sizes by extending the maximum data block size much larger than 60 bytes (e.g., 768 bytes) and decrease the granularity of the set to minimize padding inefficiencies.

The following text is proposed for the SDD:

11.7 Channel Coding for HARQ Incremental Redundancy

Figure 1 illustrates the fundamental operations in the channel coding of a CTC-encoded PDU in 802.16m. The PDU is first segmented into a number of data blocks. The size of a data block is generically referred to as *D* in this contribution. Each data block is independently encoded by a convolutional turbo code (CTC) encoder to produce 3*D* encoded bits, which is followed by a subpacket generation procedure that selects a size-*E* subset of the encoded bits to transmit over the channel. The bits of the encoded data block are used to select complex modulation symbols and are subsequently mapped to physical channel resources.

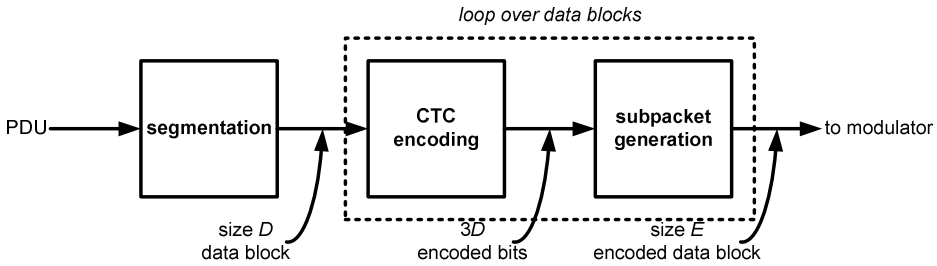

Figure 3. Fundamental operations in the channel coding of a PDU.

The data block size *D* and encoded block size *E* (the ratio of which is code rate) are treated as independent parameters.

Figure 2 illustrates the subpacket generation procedure. For a data block containing *D* bits, the convolutional turbo code (CTC) naturally generates 6 size-*D*/2 encoded bit streams labeled A, B, Y_1, Y_2, W_1, and W_2. After interleaving and interlacing, a subset of the bits is selected for transmission over the channel. For instance, the figure shows that on the first transmission, a subset of size E1 is selected for transmission over the channel. The size of the subset is chosen to exactly match the coded bit capacity of the physical channel allocation for the data block. No requirement need be imposed that the resulting code rate, *D*/*E*, be a member of a predefined set of allowable code rates.

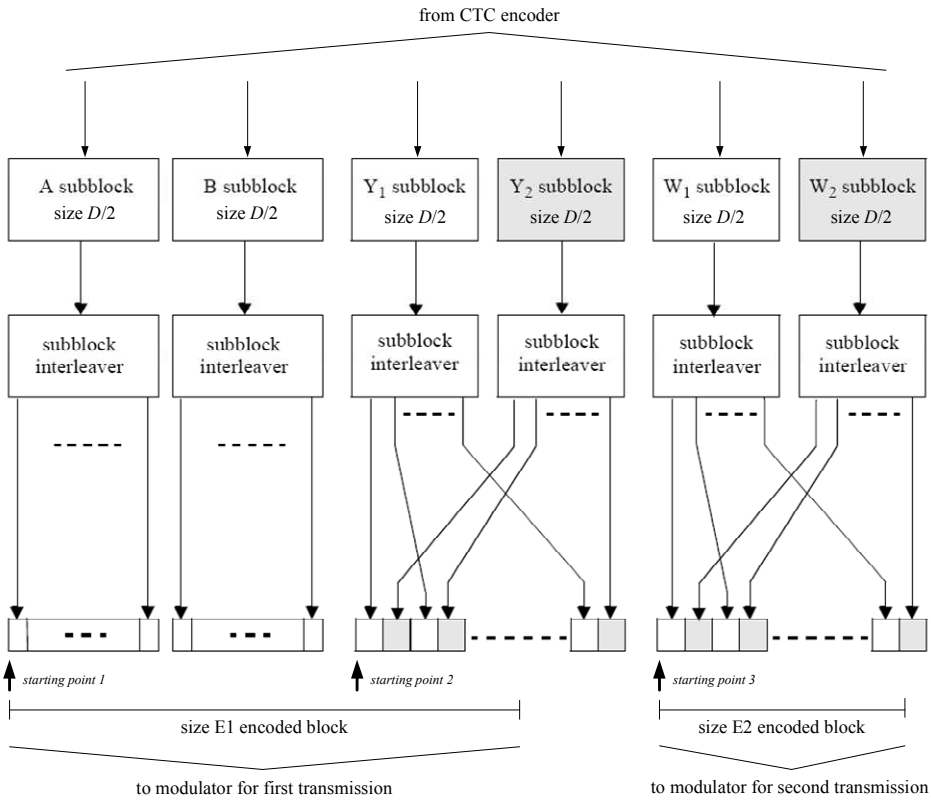

Figure 4. Subpacket generation procedure.