Title: **Remarks on 802.11-95/76 (section 10 of 802.11D1.1)**

Date: May 1995

Author: Naftali Chayat

LANNAIR
Atidim Technological Park, Bldg. 3
Tel Aviv, 61131, ISRAEL
Tel: +972-3-645-9135
Fax: +972-3-648-7146
email: naftali@lannet.com

Enclosed several comments, mostly of editorial nature. I will not be able to participate in May 95 meeting, so I pass my input in this Ad Hoc paper.

1. Inconsistency in scrambler description vs. figure remained in D1.1

The inconsistency, as pointed out in remark by A. Bolea, remained. To my opinion, the figure should be updated, and the sequence fully listed in the text, as in a draft document I passed to Dean K. (text enclosed):

The PLCP_PDU data whitener uses a length-127 frame synchronous scrambling followed by a 32/33 Bias Suppression Encoding to randomize the data from highly redundant patterns and to minimize the data DC bias and maximum run lengths. Data bytes are placed in the transmit serial bit stream LSB first and MSB last. The frame synchronous scrambler uses the generator polynomial S(x) as follows:

$$S(x) = x^7+x^4+1$$

and is illustrated in Figure 10-4. The 127-bit sequence generated repeatedly by the scramber is (leftmost used first) 00001110 11110010 11001001 00000010 00100110 00101110 10110110 00001100 11010100 11100111 10110100 00101010 11111010 01010001 10111000 1111111. The same scrambler is used to scramble transmit data and descramble receive data. The data whitening starts with the first bit of the PLCP_PDU which follows the last bit of the PLCP Header. The specific bias suppression encoding method used is defined in Figure 10-7a. The format of the packet after data whitening is as shown in Figure 10-5.

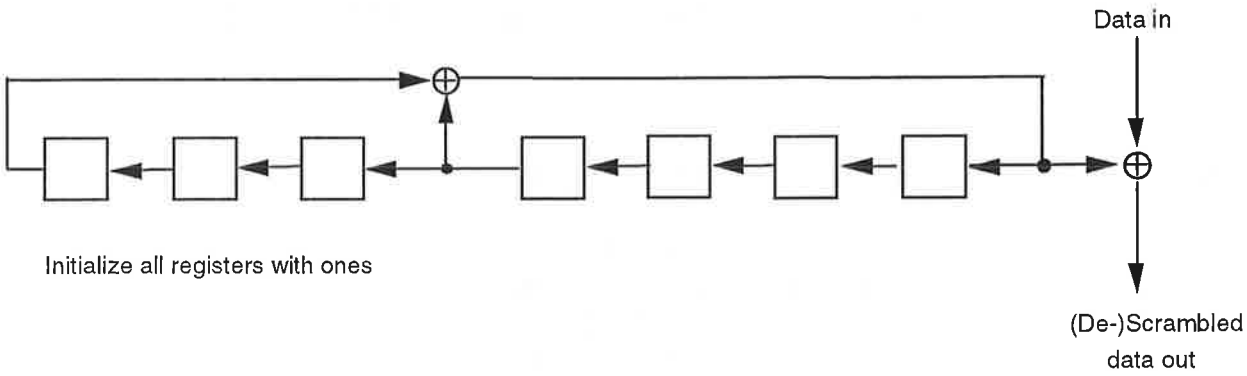

Figure 10-4: Frame Synchronous Scrambler/Descrambler.

2. Whitener decoding algorithm

The header bias calculation in decoding (fig. 10-11a) should be same as in encoding (fig. 10-7a), for consistency (and correctness). There is a factor of 2 between the two bias computation methods, because 2*b(i)-1 has values +/-1, while in fig. 10-7b binary symbols have weight +/-2.

3. Bias error monitoring

The error monitoring method presented in 10-11a is very inefficient. In order to generate an "out-of-range" violation after an error the accumulated bias should be at or near the boundary before the error occurred. The probability of the bias being at the edge of the allowable range is about 2^{-32}=2e-10 for binary data, and even less for 4-level data (this is a crude estimate, the correct one is less). There is a chance of less than 1e-7 of encountering the "edge-of-range" state in a whole packet! Therefore the chance of catching an error with "range violation" method is slight, unless there are multiple errors and they tend to be in the same direction.

A method which has a higher chance of catching a bias error is checking the polarity of **bias_next_block** vs. **accum** before summing them, i.e.:

if **bias_next_block** * **accum** >0 then report error;

accum = accum + bias_next_block;

because this violations are detected when accum is around 0, which has a probability of at least 10% (again, crude estimate).

Again, my personal opinion that this check should not be mandatory, as CRC32 check at end of packet suffices and there is no gain in detecting the error earlier.

4. TXD_UNIT and RXD_UNIT are always binary

In table 10-8 TXD_UNIT and RXD_UNIT are 0,1 for 1Mb/s and 0,1,2,3 for 2 Mb. In the text 10.5.5.1 and 10.5.5.2 it is always zero/one. Of this two inconsistent versions, I accept the one in the text. The data is always passed to/from PLCP as binary, and the packing into symbols is internal to it. What really changes is the clock rate issued by the PLCP - once or twice in each microsecond, and this should be reflected in the "when generated" paragraphs.