8-bit CRC Choices for JANUS

The current release (ver. 0.3) JANUS CRC implementation is incorrect, truncating a 16-bit CRC to 8 bits. Half a 16-bit CRC does not an 8-bit CRC make! Some preliminary testing by Giovanni Zappa has showed that the current CRC catches about 98.5% of errors in the 56-bit JANUS packet, which isn't so bad, but far from ideal. It means that 1.5% if the damaged packets are passed as error-free, which in some applications could be serious. We should therefore replace the existing CRC routine asap. Note that decoding all previous JANUS packets will return the wrong CRC result with the replacement CRC routine.

All prime 8-bit CRC polynomials will catch any consecutive sequence of bit errors up to 7 bits in length and all 2-bit errors (non-consecutive) in a packet of 255 bits or less (which JANUS is). We need a CRC algorithm to capture burst errors, rather than uniformly randomly-distributed errors, which is what the CRC is designed for, but which polynomial should we choose?

Koopman and Chakravarty [1] give a way to select the 'best' 8-bit CRC polynomial based on Hamming Distance (HD), the least number of bits in error that can result in a failure to detect the error. The HD is not necessarily the best measure of performance for our purpose, because we expect burst-like errors. After interleaving, these may be distributed non-consecutively, but the probability distribution of number of error bits may still not be smooth and is likely to be very heavy-tailed.

Based on the HD criteria, we might choose 0xEA or 0x97; both achieve a Hamming Distance (HD) of 4. On the other hand, 0x9c has a very poor HD of only 2 with a Hamming Weight (HW), (the number of ways in which the failure can occur) at that distance of 66 but only half the HW for 6 bit errors (124,248 compared to 253,084) compared to 0xEA. So would we rather save 66 possible kinds of 2-bit errors in exchange for adding 128,826 6-bit error ways to mess up? Perhaps not.

We also need to bear in mind that all obvious choices of 8-bit CRC Polynomial are likely to perform above the 99.999% rate on a 56-bit message (the length of the JANUS packet minus the CRC) so our choice may be more politically determined by what potential users are familiar and comfortable with. There is no point having the absolute 'best' mathematical/technical choice if the solution comes out a little too late and is not widely adopted (c.f. VHS versus Betamax video codecs).

Wikipedia lists the 8-bit polynomials (ignoring the highest-order bit, so that the actual polynomial has 9 terms) that have been widely adopted. We might want to choose one of these. My vote, though a weak one, is for CCIT.

Name	Polynomial	Adopted by	Representations
CRC-8-CCIT	x^8+x^2+x+1	ATM HEC, ISDN, ITU-T	0x07 / 0xE0 / 0x83
CRC-8-Dallas/Maxim	x^8+x^5+x^4+1	1-Wire bus	0x31 / 0x8C / 0x98
CRC-8	x^8+x^7+x^6+x^4+x^2+1	-	0xD5 / 0xAB / 0xEA
CRC-8-SAE J1850	x^8+x^4+x^3+x^2+1	-	0x1D / 0xB8 / 0x8E
CRC-8 WCDMA	x^8+x^7+x^4+x^3+x+1	-	0x9B / 0xD9 / 0xCD

The three representation versions in hex notation for each polynomial represent normal, reversed and reverse of reciprocal values, depending on whether little endian or big endian etc. are being used.

If someone really wanted to estimate the performance, it might not be so hard to do so by brute force. If we assume a fixed SFD and ignored the 9 'spare' bits, we are already down to 43 bits. We could test each of several CRC polynomials against all possible errors in the variable part of the packet and gather the statistics. We could then choose some pdf of how the errors occur (essentially how 'bursty' they are and how long the bursts last) and generate the metric for least undetected errors under each model for each of a limited set of polynomials. This would take some time to code up and might run over a few weekends. But life is short and I doubt it would really be worth the effort.

References

1. Koopman, Philip; Chakravarty, Tridib, "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks". The international conference on dependable systems and networks, DSN 2004. http://www.ece.cmu.edu/~koopman/roses/dsn04/koopman04_crc_poly_embedded.pdf.