Revisiting Fletcher and Adler Checksums

Theresa Maxino
Carnegie Mellon University
Pittsburgh, PA 15213
maxino@cmu.edu

Abstract

Checksums are routinely used to detect data transmission errors. However, design decisions about which checksum to use are difficult because of a lack of information about relative effectiveness of available options. We study the error detection effectiveness of the Fletcher and Adler checksums for random independent bit errors and burst errors. Our study reveals that in most cases the Fletcher checksum should be used instead of the Adler checksum.

1. Introduction

Checksums are a common way to ensure data integrity (e.g., TCP[1][8], ZLIB[2]). This paper examines two of the most commonly used checksum approaches, the Fletcher checksum and the Adler checksum, and evaluates their comparative error detection and compute cost effectiveness. Our results indicate that the Fletcher checksum should be used instead of the Adler checksum for most applications.

In the balance of this paper, we describe the Fletcher and Adler algorithms and their performance for random independent bit errors and burst errors. Error detection effectiveness is evaluated via simulation results. We give insight into the strengths and weaknesses of both checksums, with an emphasis on particular vulnerabilities to undetected errors based on data values and bit error patterns. We also examine the cost effectiveness of both, and clarify a misconception regarding their relative effectiveness.

2. Background and Related Work

A checksum is created by adding all the bytes or words in a data word to create a checksum value. In a Frame Check Sequence (FCS), the checksum is appended to the data word and transmitted with it. Receivers recompute the checksum of the received data word and compare it to the received checksum value. If the computed and received checksum match, then it is unlikely that there was an error. Of course it is possible that some pattern of altered bits in a message results in an erroneous data word matching the transmitted (and also potentially erroneous) checksum value. There is a tradeoff between the computing power used on the checksum calculation, size of the FCS field, and probability of such undetected errors.

The Fletcher checksum [3] and the later Adler checksum [2] are both designed to give error detection properties almost as good as CRCs with significantly reduced computational cost. [4] and [6] present efficient implementations for the Fletcher checksum.

Although both the Fletcher and Adler checksums have been around for years, little is published about their error detection performance. Fletcher published error detection information in his original paper [3], and [2] states that the Adler checksum is an improvement over the Fletcher checksum. However, [5] presents an analytic comparison of CRC-32, Fletcher-32, and Adler-32 (for a code word length of 8KB) that seemingly contradicts [2]. This apparent contradiction encouraged us to look deeper into this matter.

3. Effectiveness Evaluation

3.1. Methodology

Performance evaluation of both checksum algorithms was via simulated fault injection. Each experiment generated a data word with a specific data value, then computed a checksum across that data word. The resultant code word (data word plus checksum) was subjected to a specific number of bit inversion faults. The checksum of the faulty data word was then computed and compared against the (potentially also faulty) FCS value of the faulty codeword. If the FCS value of the faulty code word matched the checksum computed across the faulty data word, that particular set of bit inversions was undetected by the checksum algorithm used. Identical data word values were used for both checksums. Data word lengths were always multiples of the checksum size. The data used in each experiment varied, including random data, all zeros, all ones, and repeated data patterns.

The Hamming Distance (HD) of a checksum is the smallest number of bit errors for which there is at least one undetected case. With the assumption of random independent bit errors in a binary symmetric channel, the main contributing factor to checksum effectiveness is the fraction of undetected errors at the HD, since the probability of more errors occurring is significantly less likely. The faults injected in each experiment included all possible 1-, 2-, and 3-bit errors in the code word for each data word value examined. The number of undetected errors at the HD of a given checksum algorithm was then used to come up with the probability of undetected errors (P_{ud}) at a certain bit error rate (BER) for that particular checksum.

3.2. One's Complement Fletcher Checksum

The Fletcher checksum [3][8] is only defined for 16-bit and 32-bit checksums, but in principle could be computed for any block size with an even number of bits. We use the one's complement addition version, which provides better error detection than the two's complement addition version. This was confirmed experimentally and agrees with [4]. (Throughout this paper "Fletcher checksum" means "one's complement addition Fletcher checksum.")

A Fletcher checksum is computed with a block size that is half the checksum size (e.g., a 32-bit Fletcher checksum is computed with a block size of 16 bits across the data word). The algorithm used to compute the checksum iterating across a set of blocks D_0 to D_n is:

```
Initial values:  $sumA = sumB = 0$;
For all $i$ :  {  $sumA = sumA + D_i$;
                  $sumB = sumB + sumA$;  }
```

sumA and sumB are both computed using the same block size. The resulting checksum is sumB concatenated with sumA to form a checksum that is twice the block size. The accumulation of sumB makes the checksum sensitive to the order in which blocks are processed.

Fletcher checksum error detection properties are data-dependent. For all zero data the only undetected error is one in which all bits are changed from zeros to ones. (Recall that 0xFF also represents zero in one's complement notation.)

The Fletcher checksum has HD=3 up to a certain, modulo-dependent, code word length and HD=2 for all remaining code word lengths. All 2-bit errors are detected for data word lengths less than $(2^{k/2} - 1) * (k/2)$ bits where k is the checksum size and $(2^{k/2} - 1)$ is equal to the Fletcher checksum modulus. [3] states further that all 2-bit errors are detected provided that they are separated by fewer than $(2^{k/2} - 1) * (k/2)$ bits, k being the checksum size.

The highest P_{ud} occurs when data in each bit position of the blocks is equally divided between zeros and ones. Random data word values give approximately worst case error detection performance due to a relatively equal distribution of zeros and ones in each bit position.

The Fletcher checksum detects all burst errors less than $k/2$ bits long, k being the checksum size. It is vulnerable to burst errors that invert bits from all zero to all one since those values are both equal to zero in one's complement notation (the Adler checksum has the same vulnerability).

Figure 1 shows Fletcher checksum performance. A BER of 10^{-5} is used, assuming random independent bit inversions.

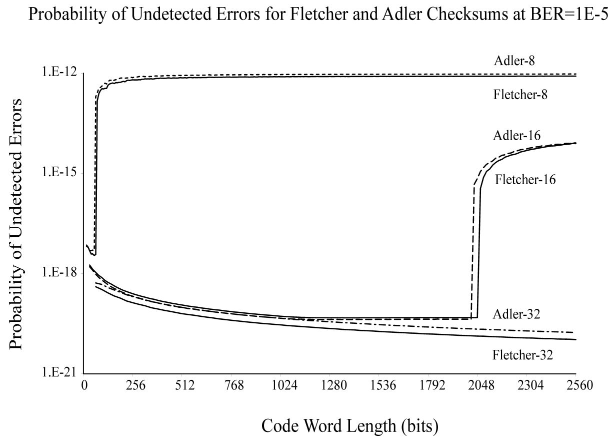

Figure 1. Fletcher and Adler Checksum Performance for Random Data (mean of 10 trials)

3.3. Adler Checksum

The Adler checksum [2] is only defined for 32-bit checksums, but in principle could be computed for any block size with an even number of bits. The Adler checksum is similar to the Fletcher checksum and can be thought of in the following way. By using one's complement addition, the Fletcher checksum is performing integer addition modulo 65535 for 16-bit blocks. The Adler checksum instead uses a prime modulus in an attempt to get better mixing of the checksum bits. The algorithm is identical to the Fletcher algorithm, except sumA is initialized to 1 and each addition is done modulo 65521 (for 32-bit Adler checksum) instead of modulo 65535.

Although the Adler checksum is not officially defined for other data word lengths, we used the largest prime integers less than 2^4=16 and less than 2^8=256 to implement 8- and 16-bit Adler checksums for comparison purposes. Having a similar algorithm, the Adler checksum has similar performance properties to the Fletcher checksum. All 2-bit errors are detected for data word lengths less than $M * (k/2)$ bits where k is the checksum size and M is equal to the Adler checksum modulus. As with the Fletcher checksum, the worst case P_{ud} is with an equal number of zeros and ones in each data block bit position, meaning that random data has nearly worst-case undetected error performance.

Adler-8 and Adler-16 detect all burst errors less than $k/2$ bits long. Adler-32 detects all burst errors up to 7-bits long. ([2] defines Adler-32 blocks to be 1 byte or 8 bits wide.)

Figure 1 shows Adler checksum performance. A BER of 10^{-5} is used, assuming random independent bit inversions.

4. One's Complement Fletcher Checksum vs. Adler Checksum

The Adler checksum has been put forward as an improvement of the Fletcher checksum [2], and it is commonly believed that the Adler checksum is unconditionally superior to the Fletcher checksum [7]. (In private communication, Mark Adler stated that what [2] meant was that Adler-32 is an improvement over Fletcher-16. At that time, he was not aware of Fletcher-32, but this point is not widely known and is not apparent in [2].)

The better mixing of bits that the Adler checksum provides due to its prime modulus has been claimed to provide better error detection capabilities than the Fletcher checksum. We have found that this is often not the case. [5] also shows that Fletcher-32 is better than Adler-32 at 8KB.

The Adler checksum outperforms the Fletcher checksum only for 16-bit checksums, and only in that checksum's HD=3 performance region. The issue is that while the prime modulus in the Adler checksum results in better mixing, there are fewer "bins" (i.e., valid FCS values) available for code words. In most cases, this reduction in bins outweighs the gains made by better mixing. Thus, the Fletcher checksum is superior to the Adler checksum in all cases except for Adler-16 used on short data word lengths. Even then the improvement in error detection effectiveness might not be worth the increase in complexity and computational cost of performing modular addition.

5. Compute Costs

Selection of the best checksum for a given application is usually not based on error detection properties alone. Other factors such as computational cost frequently come into play as well.

To determine the compute cost, we examined the actual code that was used in our experiments. We found that the Adler checksum has a compute cost that is 1.25 times that of the Fletcher checksum. This agrees with our expectations since the prime modulo used by the Adler checksum results in additional computations. Of course these performance numbers are just approximate ratios and many factors determine the best choice for a particular application.

6. Future Work

We plan to look into the performance of other commonly used checksums, namely, exclusive or (XOR), two's complement addition, one's complement addition, and cyclic redundancy codes (CRC). We also plan to re-examine commonly held notions regarding P_{ud}, and the relative effectiveness of the Fletcher and Adler checksums compared to CRCs.

7. Conclusions

The error detection properties of the Fletcher and Adler checksums are very similar. However, based on our studies of undetected error probabilities, the Fletcher checksum is often a better choice. It has a lower computational cost than the Adler checksum and, contrary to popular belief, is also more effective in most situations.

8. Acknowledgements

We would like to thank Prof. Philip Koopman for the invaluable insights, help, and guidance he has provided us. This research was supported in part by a grant from Bombardier Transportation and the General Motors Collaborative Research Laboratory at Carnegie Mellon University.

References

[1] R. Braden, D. Borman, and C. Partridge. Computing the internet checksum. Network Working Group Request for Comments (RFC) 1071, Sept. 1988.

[2] P. Deutsch and J.-L. Gailly. ZLIB compressed data format specification version 3.3. Network Working Group Request for Comments (RFC) 1950, May 1996.

[3] J. G. Fletcher. An arithmetic checksum for serial transmissions. *IEEE Transactions on Communications*, COM-30(1):247–252, Jan. 1982.

[4] A. Nakassis. Fletcher's error detection algorithm: How to implement it efficiently and how to avoid the most common pitfalls. *Computer Communication Review*, 18(5):63–88, Oct. 1988.

[5] D. Sheinwald, J. Satran, P. Thaler, and V. Cavanna. Internet protocol small computer system interface (iSCSI) cyclic redundancy check (CRC) / checksum considerations. Network Working Group Request for Comments (RFC) 3385, Sept. 2002.

[6] K. Sklower. Improving the efficiency of the OSI checksum calculation. *Computer Communication Review*, 19(5):44–55, Oct. 1989.

[7] Wikipedia. Adler-32. http://en.wikipedia.org/wiki/Adler-32.

[8] J. Zweig and C. Partridge. TCP alternate checksum options. Network Working Group Request for Comments (RFC) 1146, Mar. 1990.