Analysing the Efficiency and Performance of CHECKSUM in High-Speed Data Networks

1. Introduction

In modern high-speed data networks, the efficient and accurate transmission of data is paramount. To ensure the reliability of transmitted information, robust error detection and correction mechanisms are essential. This project delves into a detailed analysis of the Cyclic CHECKSUM mechanism, a widely used error detection algorithm, in the context of high-speed data networks.

The basic idea behind a checksum is to generate a fixed-size value by performing a mathematical operation on the data bits. This value, the checksum, is then appended to the original data before transmission. The recipient can then use the checksum to detect errors that may have occurred during transmission. While checksums can't correct errors, they are effective at detecting them, allowing for retransmission or other error-handling mechanisms.

2. CHECKSUM Fundamentals

2.1 Conceptual Understanding

A CHECKSUM is a mathematical value that is calculated from a block of data and is used to detect errors that may have been introduced during transmission or storage. It is a simple and effective error-detection technique that is widely used in networking protocols, file storage, and data transmission scenarios where maintaining data integrity is crucial.

The characteristics of a checksum are that it should be small enough to add minimal overhead but large enough to provide sufficient error-detection capabilities. Different checksum algorithms use various mathematical operations, such as addition, bitwise XOR, or modular arithmetic.

2.2 Mathematical Background

1's Complement Checksum for 8-Bit Words:

1. **Calculation:**
   - Compute the sum of all the 8-bit words in the message.

$$Sum = word1 + word2 + \ldots + wordN$$

Fold Carry Bits:

- Fold any carry bits back into the sum until no carry remains.

Sum=(SumAND0*xFF*)+(SumSHIFT RIGHT8)

2. **1's Complement:**
    - Calculate the 1's complement of the sum.

1s_Complement_Checksum= ~ (Sum)

3. **Result:**
    - The result is an 8-bit value that can be appended to the original message for transmission.

Fletcher's Checksum:

Fletcher's checksum is a simple checksum algorithm that involves summing the data and its sum to provide error detection.

1. **Initialization:**
    - Initialize two 8-bit variables, sum1 and sum2, to zero.
2. **Calculation:**
    - Iterate over each byte in the message and update the sums.

sum1=(sum1+byte)%255
sum2=(sum1+sum2)%255

3. **Result:**
    - The result is a pair of 8-bit values (sum2,sum1) that can be appended to the original message for transmission.

Mathematically, Fletcher's checksum can be expressed as follows:

Fletcher_Checksum=(sum2,sum1)

These checksums are relatively simple and are often used for basic error detection in scenarios where a lightweight approach is sufficient. It's important to note that these checksums are not designed for error correction but can effectively detect common errors in transmitted data.

3. Implementing CHECKSUM in High-Speed Data Networks

Checksum Calculation at Sender:
The sender calculates the checksum for the entire message by performing a mathematical operation on the data bits, resulting in a fixed-size value. This value is then appended to the original data before transmission.

Checksum Appending:
The calculated checksum is appended to the original message or data packet by concatenating it to the end of the message.

Transmission Process:

The overall process of sending the data packet (data + checksum) over the network involves the following steps:

1. The sender sends the data packet over the network to the recipient.
2. The recipient receives the data packet and extracts the data and checksum from the packet.
3. The recipient calculates the checksum of the received data.
4. The recipient compares the calculated checksum with the received checksum to detect errors.

Reception at Receiver:

Upon receiving the data packet, the recipient performs the following steps:

1. The recipient extracts the data and checksum from the packet.
2. The recipient calculates the checksum of the received data.
3. The recipient compares the calculated checksum with the received checksum to detect errors.

Error Detection:
The role of checksums in detecting errors is to compare the received and calculated checksums. If the two checksums match, the data is assumed to be error-free. If the checksums do not match, an error is detected, and the recipient can request the sender to retransmit the data.

Error Handling:

Error handling using checksums involves requesting the sender to retransmit the data if an error is detected. The recipient can send a message to the sender requesting the retransmission of the data. The sender can then retransmit the data packet, and the recipient can perform the error detection process again to ensure that the data is error-free.

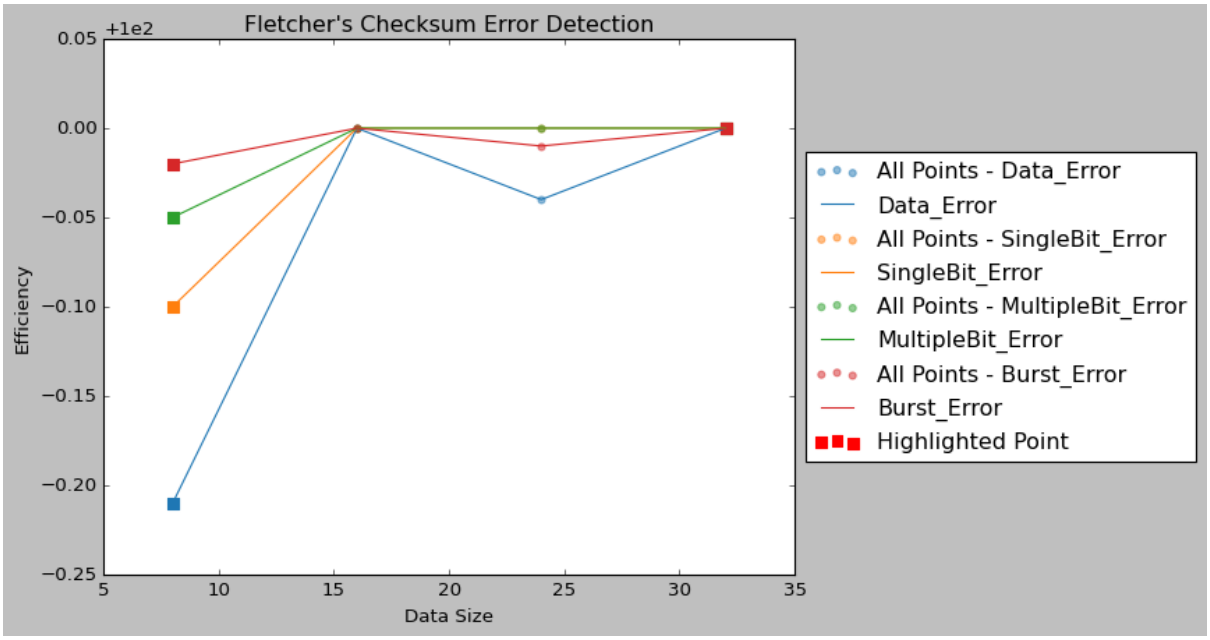
4. Performance Analysis

1. **Overview of Checksum:**
    - Checksum is an error detection method used by upper layer protocols and is considered more reliable than LRC, VRC, and CRC.
2. **Algorithm Description:**
    - Checksum involves a simple and fast algorithm where the sender's checksum generator divides the data into equal subunits of n-bit length.
    - These subunits are added using the one's complement method, and the resulting sum is complemented to obtain the checksum.
    - The checksum is then appended to the original data unit and transmitted to the receiver.
3. **Receiver Verification:**
    - The receiver, after receiving data + checksum, passes it to a checksum checker.
    - The checker divides the data unit into equal subunits, adds them, and complements the result.

- If the complemented result is zero, the data is considered error-free; otherwise, it is rejected.

4. **Error Detection Capability:**
   - The checksum detects all errors involving an odd or even number of bits.
5. **Limitation in Error Detection:**
   - A limitation arises when one or more bits of a subunit are damaged, and the corresponding bits of opposite value in another subunit are also damaged, leaving the sum of those columns unchanged.
6. **Checksum Size and Data Transmission Efficiency:**
   - The size of the checksum can impact data transmission efficiency.
   - Smaller checksums introduce less overhead but might have limitations in detecting certain types of errors.
7. **False Positives and Negatives:**
   - False positives occur when the algorithm incorrectly identifies errors, and false negatives occur when the algorithm misses actual errors.
8. **Checksum Collisions:**
   - Checksum collisions can occur when different data payloads produce the same checksum, potentially leading to undetected errors.

Comparing the performance with changing length of data


Fletcher's Checksum Error Detection
+1e2
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
Efficiency
5
10
15
20
25
30
35
Data Size
All Points - Data_Error
Data_Error
All Points - SingleBit_Error
SingleBit_Error
All Points - MultipleBit_Error
MultipleBit_Error
All Points - Burst_Error
Burst_Error
Highlighted Point