

NEW STOP & WAIT ARQ PROTOCOL

Nitin Jain, Rishi Asthana & Manuj Darbari

Uttar Pradesh Technical University, Lucknow nitinjain_22@rediffmail.com,
asthana_rishi@yahoo.com, manujuma@rediffmail.com

ABSTRACT

In all types of data communication systems, errors may occur. Therefore error control is necessary for reliable data communication. Error control involves both error detection and error correction. Previously error detection can be done by Cyclic Redundancy Check (CRC) codes and error correction can be performed by retransmitting the corrupted data block popularly known as Automatic Repeat Request (ARQ). But CRC codes can only detect errors after the entire block of data has been received and processed. In this work we use a new and “continuous” technique for error detection namely, Continuous Error Detection (CED). The “continuous” nature of error detection comes from using arithmetic coding. This CED technique improves the overall performance of communication systems because it can detect errors while the data block is being processed. We focus only on ARQ based transmission systems. We will show have the proposed CED technique can improve the throughput of ARQ systems by up to 15%.

Keywords: Cyclic redundancy check codes, arithmetic coding, automatic repeat request.

1 INTRODUCTION

When we talk about any type of data communication system, we concern only on its reliability. In all types of data communication systems, errors may occur. Error control is the only way out for avoiding this problem. It comes by detecting the error in first step and then correcting it in another step. For error detection we had CRC codes and for error correction we use to retransmit the corrupted data which is popularly known as ARQ. Although efficient, CRC's can detect errors only after an entire block of data has been received and processed. An error detection scheme that is “continuous” can detect errors while the block is being processed. Thus, it can enhance the overall performance of the communication systems.

In this paper, we use this type of new and continuous method for detecting the errors. The new method of error detection is based on the arithmetic coder, and allows for an efficient way to detect errors continuously in the bit-stream by investing a controlled amount of redundancy in the arithmetic coding operation. During our research, we became aware that the idea of continuous error detection based on arithmetic coding had been tackled earlier by Boyd *et al.* [1], albeit with little system performance analysis, an exposition of its utility in communication systems. In this paper, we not only undertake a more rigorous analysis of this paradigm, quantifying the underlying tradeoffs involved in the process, but also establish impressive gains in system performance attainable through sophisticated

integration of this novel paradigm into popular, powerful transmission scenarios such as ARQ.

Upon applying this method of error detection to stop-and-wait ARQ, gains in throughput were achieved over conventional ARQ schemes at all bit error probabilities. Result shows that the throughput of new stop-and-wait ARQ protocol i.e. with CED is approximately 15% enhanced than the throughput of the conventional stop-and-wait ARQ protocol i.e. with CRC.

The rest of the paper is organized as follows. The basic idea behind the continuous error detection is introduced in Section 2. Section 3 presents an application of CED for ARQ transmission where it provides significant throughput gains over conventional CRC-based schemes. We conclude in Section 4. References are given in Section 5.

2 IDEA BEHIND CED

To understand the error detection scheme, an understanding of how the arithmetic coder works is necessary. We assume that the reader is familiar with arithmetic coding and refer readers that are unfamiliar with arithmetic coding to [2].

Arithmetic coding is a method of data compression in which data strings are mapped to code strings which represent the probabilities of the corresponding data strings. The method in which this mapping is achieved requires a model which specifies the assumptions that are made about the source data. A simple example of a model is the memoryless model where the current symbol being

encoded is independent of the previously encoded symbols. Another simple example is the first-order Markov model, where the current symbol being encoded is dependent only on the previously encoded symbol. For simplicity, we will examine the memoryless model, keeping in mind that the analysis generalizes to more sophisticated models. Thus, encoding and decoding via the arithmetic coder function by repetitively partitioning subintervals within the unit interval $[0, 1)$ according to the probabilities of the data symbols.

The basic idea is simple and consisting of adding a “forbidden” symbol that is never encoded by the arithmetic coder, but nonetheless occupies a nonzero measure on the set $[0, 1)$, then upon decoding, if an error occurs, this “forbidden” symbol is guaranteed to be decoded due to the loss of synchronization. The amount of time it takes to decode the “forbidden” symbol after the occurrence of an error is inversely related to the amount of redundancy added through introducing the “forbidden” symbol. This allows for control of the number of bits we suspect need to be retransmitted. In fact, we can guarantee to a specified accuracy, that errors will be localized to the previous m bits (where m is a function of the amount of redundancy added) upon decoding the “forbidden” symbol. This is useful in an ARQ setting, because as soon as the error is detected, we have a statistical assurance as to how many bits need to be retransmitted to ensure that the bit in error will be retransmitted.

3 APPLICATION OF CED

Simulations were run using a binary symmetric channel at various bit-error probabilities. Several ten kilobit packets were sent at each bit-error probability, and the resulting throughput was calculated. As a measure of performance, we compared our method of ARQ i.e. with CED to the conventional methods of ARQ i.e. with CRC. The conventional methods of ARQ function by dividing the data into packets and then attaching CRC's [3] to each packet. Upon detection of an error in the conventional ARQ method, a retransmission of the entire block is requested. To simulate a fair comparison for our method versus conventional ARQ methods, we used the optimal packet size for each bit-error probability tested using conventional ARQ. The optimal packet size was calculated by differentiating the throughput equation for conventional ARQ (details can be found in [4]) with respect to the packet size, and solving for the packet size which maximizes throughput. The resulting throughputs are shown in Fig. 1. and we see that the new method of ARQ outperforms conventional ARQ methods at all bit-error probabilities.

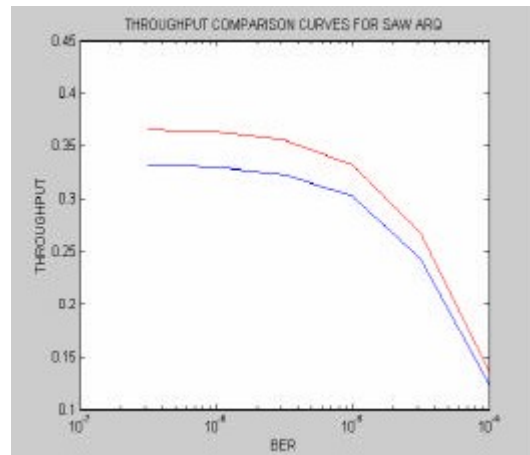


Figure 1: Throughput comparison curves for new stop-and-wait ARQ protocol i.e. with CED (upper curve in red color) versus conventional stop-and-wait ARQ protocol i.e. with CRC (lower curve in blue color).

4 CONCLUSION

In this paper we have introduced a new method of error detection for common ARQ protocols. We analytically characterized the tradeoff of added redundancy versus error-detection capability and formulated a method for incorporating this new error detection “device” into an ARQ type scenario.

We would also like to mention here that CED can be put to good use to improve throughput performance of transport protocols like TCP over heterogeneous networks, where early detection of an error can result in a potentially greater number of retransmits, thereby increasing the probability of successful reception over a fading channel. This is currently being verified. The goal of this work is to present the benefits that communication systems can derive from using CED for throughput enhancement.

5 REFERENCES

- [1] C. Boyd, J. Cleary, S. Irvine, I. Rinsma-Melchert, and I. Witten, “Integrating error detection into arithmetic coding,” *IEEE Trans. Commun.*, vol. 45, pp. 1–3, Jan. 1997.
- [2] G. Langdon, “An introduction to arithmetic coding,” *IBM J. Res. Develop.*, vol. 28, pp. 135–149, Mar. 1984.
- [3] T. Ramabadran and S. Gaitonde, “A tutorial on crc computations,” *IEEE Micro*, vol. 45, pp. 62–74, Aug. 1988.
- [4] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*. Reading, MA: Addison-Wesley, 1987.