An Efficient Software Implementation of a Forward Error Correcting Code*

Jean-Charles Henrion

Institut d'électricité Montefiore B.28, Université de Liège, B-4000 Liège, Belgium
Email: henrion@montefiore.ulg.ac.be

Abstract. Today, Forward Error-Correcting (FEC) codes are mainly implemented in hardware, and many believe that their complexity prohibit their software implementation. This paper presents in detail how the performances of a software implementantion can be significantly improved. Different levels of optimization which are independent of the working environment are presented and discussed. The coding throughput of 100 Mbps on an UltraSparc 1 shows that FEC codes can be easily added to multimedia applications without requiring dedicated hardware support. As a case study, we use FEC codes to protect AAL5-PDUs from cell losses in ATM networks.

1 Introduction

Communication systems usually rely on two types of error-correction mechanisms. The ARQ (Automatic Repeat Request) techniques, which require data retransmission, are used in many data applications and protocols. The FEC (Forward Error Correction) techniques rely on redundancy in the transmitted messages to recover errors at the receiving side. Three types of applications use FEC rather than ARQ techniques: applications that are sensitive to the delay (e.g. real time systems), Multicast applications (e.g. multimedia) and applications which cannot retain information for a long time (e.g. satellite). The FEC codes are used to protect information messages which could be altered, or even partially deleted. Each message is segmented into a set of k independent data blocks which are transmitted as n encoded data blocks (k ≤ n). The receiver will be able to restore the information message if it has received at least k of the n data blocks without errors. Although our coder (decoder) is specific to ATM Networks, the optimized coding and decoding procedures could be easily re-used in another context.

2 A FEC code for ATM Networks

Compared with packet-based networks such as Internet, one of the major characteristics of the ATM networks [1] is that the unit of information transfer is

* This work was partially supported by the European Commission within the ACTS AC051 OKAPI project.
a 53-byte ATM cell. As most users prefer to be able to send variable length packets, the ITU-T and the ATM forum have defined AAL5 and AAL3/4 [2]. Today, most ATM adapters provide efficient support for AAL5, and AAL3/4 is becoming less and less used. For this reason, we choose to design our FEC code above AAL5. The CS and SAR sublayers of AAL Type 5 are summarized in figure 1.

2.1 AAL Type 5

![Diagram of AAL Type 5](image)

**Acronyms**
- **AAL**: ATM Adaptation Layer
- **CS**: Convergence Sublayer
- **SAR**: Segmentation And Reassembly sublayer
- **SDU**: Service Data Unit
- **PDU**: Protocol Data Unit
- **CPCS**: Common Part CS
- **SSCS**: Service Specific CS

**Fig. 1. The AAL Type 5**

In ATM networks, cell losses may occur for several reasons. First, bit errors in the ATM cell header may force the discarding of the whole cell if the Header Error Control (HEC) field is unable to correct these errors. Another source of cell losses is the possible congestion in the ATM switches. When cell losses occur with AAL5, we have to distinguish between two different cases. If the last cell of a CPCS-PDU is not lost, the receiver will detect an error because the number of received cells does not agree with the LEN field (length of the CPCS-SDU). If the last cell of a CPCS-PDU is lost, the SAR sublayer of the receiver will deliver the received cells concatenated with the cells of the next CPCS-PDU(s) upon reception of the last cell of the next CPCS-PDU(s). We also consider the case where ATM cells are misinserted in the cell flow. This rare situation [4] happens when the HEC field is unable to detect an error in the ATM cell header. We assume that the cell loss rate is relatively low and burst losses are sufficiently limited. This corresponds to the CBR and VBR categories defined by ATM forum [3].