REIHE INFORMATIK
9/93

Adaptable Forward Error Correction for Multimedia Data Streams

B. Lamparter, Otto Böhrer, W. Effelsberg, and V. Turau
Universität Mannheim
Seminargebäude A5
D-68131 Mannheim

Adaptable Forward Error Correction for Multimedia Data Streams

Bernd Lamparter, Otto Böhrer
Praktische Informatik IV
University of Mannheim
68131 Mannheim
Germany
Email: lamparter@pi4.informatik.uni-mannheim.de

Wolfgang Effelsberg*, Volker Turau†
International Computer Science Institute
1947 Center Street
Berkeley, California 94707
U.S.A.
Email: effelsberg@pi4.informatik.uni-mannheim.de
turau@prfhfb.fh-friedberg.de

Abstract

The error handling method in traditional communication protocols is error detection and retransmission. This method is inappropriate for distributed multimedia systems for two reasons: It introduces variable delay unacceptable for isochronous streams, and it is very inefficient and difficult to use in the multicast environment typical for many multimedia applications. We propose AdFEC, an adaptable Forward Error Correction scheme based on binary polynomial algebra. It produces an adaptable amount of redundancy allowing different packet types to be protected according to their importance. The scheme was implemented in the framework of the XMovie project and proved to be very efficient.

*Permanent address: Praktische Informatik IV, University of Mannheim, 68131 Mannheim, Germany
†Permanent address: Fachhochschule Gießen-Friedberg, Wilhelm-Leuschner-Str. 13, 61169 Friedberg

1 Introduction

Two different approaches can be used to correct transmission errors in computer networks: error detection and retransmission of damaged packets (ARQ, Automatic Repeat Request), or correction of bit errors by means of redundant information (FEC, Forward Error Correction). Traditional network protocols (HDLC, ISO/OSI-TP4, TCP/IP, etc.) all work with error correction by retransmission, a solution favored for several reasons:

1. Error detecting codes require less redundancy than error correcting codes and thus save bandwidth.
2. Error detecting codes require a lower computational effort.
3. Retransmission must be implemented anyway in order to recover from loss of complete packets.

Conversely, the major advantage of FEC (Forward Error Correction) lies in its maintenance of isochronous flows. Whereas retransmission of damaged packets introduces considerable delay jitter, the computational overhead for FEC is the same for damaged as for undamaged packets. This is of particular importance for multimedia data streams in high speed networks [1, 2]. Since damaged data packets are corrected on the fly, time-consuming retransmission is never necessary. Also, multimedia data streams require high-bandwidth networks anyway, so increased redundancy can be easily accommodated.

Bit errors rarely occur in modern networks, especially if these are based on fiber optics. The main source of errors is packet loss in the switches. Current FEC procedures that focus on the correction of bit errors do not solve this problem.

A popular application of forward error correction is in CD players due to their high rate of error in reading and the errors imprinted during manufacture memory [5], and most recently in video coding [6]. The procedures serve primarily to correct individual error bits. Very few articles address the problem of reconstructing lost packets [12, 10, 1, 11]. These articles deal with packet loss in ATM networks. All packets in the data stream are protected by means of the same method, and with the same redundancy.

This article introduces a method capable of assigning different priorities to different parts of the data stream. The amount of redundancy for FEC is chosen according to the priority. We call the method AdFEC (Adaptable Forward Error Correction).

In Chapter 2 the requirements placed by isochronous data streams on error correction methods are listed. Chapter 3 describes the generation of redundant information in our

system. In Chapter 4 the error correction probabilities are calculated and Chapter 5 reports on our experiences with forward error correction within the XMovie-Project at the University of Mannheim.

2 Requirements for an Error Correction Method for Isochronous Streams of Data

A data stream is characterized as *isochronous* if the variance in the transmission delay of individual packets is close to zero. The most important requirement is the maintenance of the temporal relationship between the individual packets, i. e. it is absolutely necessary to display the data at the correct time. This applies to digital films and even more to audio streams. Isochronous streams of data are thus of major importance in multimedia applications. The human ear is much more sensitive to fluctuations in tone than the human eye to slight fluctuations in the display of images. In addition parts of images can be lost; this loss will be registered as only slightly irritating. Whereas slight fluctuations in delay can be compensated by buffering, the range of tolerance by no means extends to include retransmission. Errors must be corrected without retransmission.

A digital data stream for a movie or for audio contains more than just the digitized image/audio contents. It also contains information that must not be lost under any circumstances, such as control instructions, format data or changes in the color lookup table. Typically a higher error rate can be tolerated for content parts than for control parts, but all packets have to arrive on time.

In summarizing it can be stated that multimedia data streams can be reduced to segments of varying priority, for example:

Priority 1: Segments that may not be lost under any circumstances (e. g. control and format information as well changes in the color lookup table)

Priority 2: Segments whose loss clearly adversely affects quality (e.g. audio)

Priority 3: Segments whose loss is slightly damaging (e. g. pixel data in a video data stream)

For none of the three priorities is retransmission a tenable option. Starting from an already low rate of loss, third-priority packets can be transmitted without protection, second-priority packets should be protected by means of FEC with minimum redundancy, and first-priority packets by means of FEC with high redundancy.

3 Creating Redundancy for Forward Error Correction

3.1 Requirements for a Forward Correction of Errors

Traditional error-correcting codes (e. g. Reed-Solomon) were designed for the detection and correction of bit errors [9]. Since the demand now exists to also restore entire packets, new codes must be found. Specifically, errors need no longer be located, the lost bits are known. A feature of traditional error-correcting codes is their ability to locate damaged bits. The price of this feature is a great deal of redundancy. At issue here is to devise a code that restores the lost packets at a known error location.

Example 1: Two packets p_1 and p_2 are to be sent. A redundancy of 100% is taken into account, i. e. two additional packets may be generated. These additional packets are sent together with the original packets. In the event of packet loss, the original packets p_1 and p_2 must be restored from the remaining packets (see Figure 1). In this case two operations (labeled $\circ$ and $\bullet$) are necessary, with whose help the redundant packets can be generated.

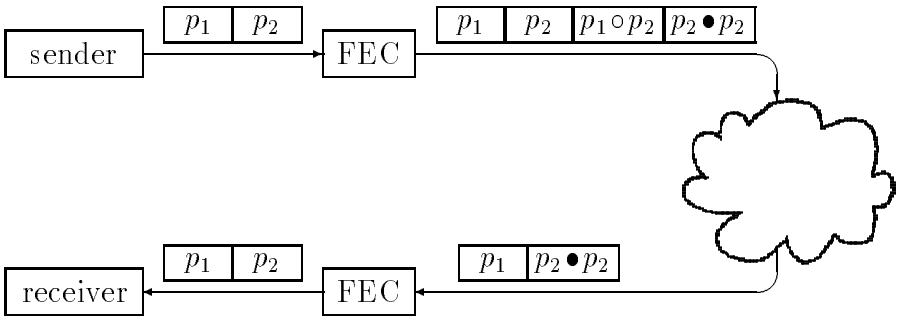

Figure 1: Principle of Forward Error Correction for Packet Losses (assume the network loses the packets p_1 and $p_1 \circ p_2$)

We can now define the problems in mathematical terms.

Definition: Bit sequence $\mathbb{B}_l = \{0,1\}^l$, for fixed $l \in \mathbb{N}$.

given: n packets $P = \{p_1, p_2, \ldots, p_n \in \mathbb{B}_l\}$

fixed: $n+m$ packets $Q = \{q_1, q_2, \ldots, q_{n+m} \in \mathbb{B}_l\}$ such that upon the arrival of at least n packets out of Q all packets of the set P can be restored.

Example 2: $n = 2, m = 1$:

Choose $q_1 = p_1, q_2 = p_2, q_3 = p_1 \oplus p_2$

$\oplus$ is the operator for binary exclusive or (XOR). If, for example, q_1 is lost, p_1 can be restored with $p_1 = q_2 \oplus q_3$.

A total of 16 binary operators can be defined combining two bits, namely all possible combinations of zeroes and ones in a four-row value table. In order to construct our packets for Q, operators are required whose result can be used for the unambiguous reconstruction of any bit fields. The only binary operator suitable for this is the XOR-operator and its negation, the equivalence. Therefore on the basis of just the 1-bit operators only one redundancy packet can be generated, all other packets would not allow an unambiguous reconstruction in all cases.

In order to generate additional independent packets, a field containing 2^n elements ($n > 1$) must be sought. Such fields can be generated with the aid of polynomial algebra [9]. Table 1 shows the operations + and $*$ for a field containing four elements.

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

$*$	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

Table 1: The operations + and $*$ for a field containing four elements

For $n = m = 2$ the operators $\circ$ and $\bullet$ can, for example, be defined as follows (the field is labeled GF for Galois Field):

$$a, b \in \mathrm{GF}(2^2) \Rightarrow \qquad a \circ b = a + b$$
$$a \bullet b = a + 10 * b$$

The two redundancy packets are thus defined as

$$c = a + b$$

and

$$d = a + 10 * b$$

In the case of loss of two data packets, the calculations in Table 2 are to be carried out.

arriving Packets	Calculations for reconstructing a	b
a, b	-	-
a, c	-	$a + c$
a, d	-	$11 * (a + d)$
b, c	$b + c$	-
b, d	$d + 10 * b$	-
c, d	$11 * c + 10 * d$	$10 * (c + d)$

Table 2: Calculations for Reconstruction of Lost Packets

Returning to our notation above we set $p_1 = a, p_2 = b, q_1 = c$ and $q_2 = d$.

Thus, if any two packets are lost, the contents of p_1 and p_2 can be reconstructed from the arriving packets.

Generally speaking, fields containing p^n elements (p is prime, $n > 0$) can be derived. The steps necessary for the construction of fields containing 2^n elements follow, prefaced by a brief introduction to polynomial algebra. A more precise derivation as well as the corresponding proofs can be found in [9].

3.2 Introduction to Polynomial Algebra

The following provides a general definition of groups and fields, followed by a presentation of special fields in polynomial algebra. These fields (Galois Fields) are then used for the forward correction of errors. Readers familiar with polynomial algebra can skip this section.

Basic Definitions

Let $\mathbb{M}$ be a nonempty set of elements and $\circ$ an inner operation that unambiguously assigns to each pair a, b of elements in $\mathbb{M}$ an element $c = a \circ b \in \mathbb{M}$.

Definition 1 *A set M together with a operation $\circ$ is a group, if the following is valid:*

1. *The operation $\circ$ is associative,*
2. *M contains a neutral element e, such that $\forall a \in M : a \circ e = e \circ a = a$*

3. $\forall a \in \mathbb{M} \;\; \exists a' \in \mathbb{M} : a \circ a' = a' \circ a = e$

Furthermore, a group is commutative if the following is true: $\forall a, b \in \mathbb{M} : a \circ b = b \circ a$. The number of elements in a group is called *order*. If the order is finite, one can also speak of a finite group.

The field can now be defined based on the group. Let $\mathbb{K}$ be a set of elements with two inner operations, addition ($+$) and multiplication ($*$):

Definition 2 *$\mathbb{K}$ is a field if the following is true:*

1. *$\mathbb{K}$ is a commutative group with regard to the addition. The element 0 is the neutral element of the addition (zero element).*
2. *$\mathbb{K}\backslash\{0\}$ is a group with regard to the multiplication. The element 1 is the neutral element with regard to the multiplication (unit element).*
3. *The multiplication is distributive over the addition, i. e. $a * (b + c) = a * b + a * c$.*

By convention, the inverse element of the addition is labeled $-a$ and the inverse element of the multiplication is labeled a^{-1}. Since in forward error correction only finite fields are of interest, from here on only such fields will be considered.

The smallest field is the binary field, represented by the set $\mathbb{K} = \{0, 1\}$ and the operators $+$ and $*$ (see Table 3, as to the non-ambiguity of the binary field see [4]). This field is also

$+$	0	1
0	0	1
1	1	0

$*$	0	1
0	0	0
1	0	1

Table 3: Addition and Multiplication in the Binary Field

referred to as GF(2) (Galois Field with the cardinality 2). Generally speaking, there exists for every prime number p a field containing p elements GF(p). Furthermore, from every GF(p) an extension can be derived: GF(p^n). It can also be shown that the cardinality of every finite field is a power of a prime number [4].

Generally the field GF(p^n) contains the elements $\sum_{i=0}^{n-1} a_i \alpha^i, a_i \in \mathrm{GF}(p)$. For example $1 + \alpha^3 + \alpha^7$ is an element out of GF(2^8). The addition can be very simply defined: Let $a, b \in \mathrm{GF}(p^n)$, and $a = a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1}$; b correspondingly; $a + b := a_0 + b_0 +$

$(a_1 + b_1)\alpha + \cdots + (a_{n-1} + b_{n-1})\alpha^{n-1}$. The addition of the respective components follows in congruence with mod p. In the case $p = 2$, the addition hence becomes the XOR-operation (see Table 4).

The definition of the multiplication is more difficult. If the usual polynomial multiplication is used, elements result that at first glance no longer belong to the field. If, for example, one has the field GF(2^4), and the elements $1 + \alpha^3$ and α^2 are multiplied in the usual manner, the result is $\alpha^2 + \alpha^5$. Hence the elements of the field are only the representatives of their equivalence class. With the aid of a generator polynomial the equivalence classes are formed, and the representatives can be found.

Definition 3 *A generator polynomial $q(X)$ of the class GF(p^n) is a polynomial of degree n over GF(p): $q(X) = \sum_{i=0}^{n} a_i X^i, a_i \in GF(p)$ for $i = 0, \ldots n$ and $a_n \neq 0$. α is a zero of this polynomial, i. e. $q(\alpha) = \sum_{i=0}^{n} a_i \alpha^i = 0$*

Thus powers of degree greater than $n - 1$ can be transformed into powers of lower degree.

Example 3: GF(2^2),
Elements of the field: GF= $\{0, 1, \alpha, 1 + \alpha\}$
generator polynomial: $p = 1 + \alpha + \alpha^2 = 0. \Rightarrow \alpha^2 = 1 + \alpha$
Derivation of the Multiplication Table:

1. The multiplication by 0 resp. 1 is trivial.
2. $\alpha * \alpha = \alpha^2 = 1 + \alpha$
3. $\alpha * (1 + \alpha) = \alpha + \alpha^2 = 1 + \alpha + \alpha = 1$
4. $(1 + \alpha) * (1 + \alpha) = 1 + \alpha + \alpha + \alpha^2 = 1 + \alpha^2 = \alpha$

All possible multiplications of GF(2^2) using the generator polynomial $p(\alpha) = 1 + \alpha + \alpha^2$ are shown in Table 5.

Not all polynomials of degree n are admissible generator polynomials. For a given polynomial it must be tested whether the conditions attached to a field are satisfied (see Definition 2). For example, the polynomial $1 + \alpha^2$ is not a generator polynomial for GF(2^2) since $(1 + \alpha) * (1 + \alpha) = 0$, which would imply that $(1 + \alpha)$ is a zero divisor of GF(2^2).

By means of suitable generator polynomials then, fields with the cardinality p^n can now be constructed. Of particular interest to the computer scientist are of course those with $p = 2$. For this reason only such fields will be considered in the following. The book by Lin and Costello [9] provides in its appendix a list of the possible generator polynomials for GF(2^n) to $n = 10$.

$+$	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

Table 4: Addition in GF(2^2)

$*$	0	1	α	$1+\alpha$
0	0	0	0	0
1	0	1	α	$1+\alpha$
α	0	α	$1+\alpha$	1
$1+\alpha$	0	$1+\alpha$	1	α

Table 5: Multiplication in GF(2^2)

3.3 The Power of Polynomial Algebra for Forward Error Correction

The fields constructed in the preceding section are to be examined regarding their power for generating redundant packets. Of particular interest is the number of independent packets that can be generated. Given the two packets a and b, generation occurs in the form $\lambda a + \mu b$, with $\lambda, \mu, a, b \in \mathrm{GF}(2^n)$.

Thus, given a field GF(2^n) and a number of packets $a_1, \ldots, a_k$, the question is: How many independent packets l can be generated by the given field from k packets? The more packets generated and transmitted, the greater the probability that the recipient will be able to reconstruct the packets sent from those that arrived. It has already been explained in Chapter 3 that by means of a field GF(2) only one single additional packet can be generated, independent of the number of given packets. The extension to the field GF(2) apparently is of little use in this respect. For this reason the fields GF(2^n) are examined.

Generally the contents of the output packets for two input packets a and b adheres to the following scheme:

Output Packets:

$p_{ij} = \lambda * a + \mu * b$ with $i, j = 1 \ldots 2^n$ and $\lambda, \mu \in \mathrm{GF}(2^n)$ (see Table 6)

Some of the packets generated in this manner are linearly dependent and their transmission thus affords no advantage for reconstruction. The transmission of the first packet is obviously useless because it is always equal to zero. The pairs of linearly dependent packets can now be combined into classes. $n-1$ elements belong to every class, in the example three elements of the table. The result yielded here is the set $\mathbb{R} = \{\{b, \alpha b, (1+\alpha)b\}, \{a+b, \alpha a+\alpha b, (1+\alpha)a+(1+\alpha)b\}, \{a+\alpha b, \alpha a+(1+\alpha)b, (1+\alpha)a+b\}, \{a+(1+\alpha)b, \alpha a+b, (1+\alpha)a+\alpha)b\}, \{a, \alpha a, (1+\alpha)a\}\}$

1.	$0+$	0	
2.	$0+$	b	
3.	$0+$	αb	
4.	$0+$	$(1+\alpha)b$	
5.	$a+$	0	
6.	$a+$	b	
7.	$a+$	αb	
8.	$a+$	$(1+\alpha)b$	
9.	αa	$+$	0
10.	αa	$+$	b
11.	αa	$+$	αb
12.	αa	$+$	$(1+\alpha)b$
13.	$(1+\alpha)a+$		0
14.	$(1+\alpha)a+$		b
15.	$(1+\alpha)a+$		αb
16.	$(1+\alpha)a+$		$(1+\alpha)b$

Table 6: Generateable Packets in GF(2^2)

Generally the following applies:

Theorem 1 *In a field GF(2^n), and for two packets a and b, exactly 2^n+1 pairs of linearly independent packets in the form $\lambda a + \mu b$, whereby $\lambda, \mu \in GF(2^n)$, can be generated.*

Remark: From every packet in the form $\lambda a + \mu b \neq 0$, $2^n - 1$ linearly dependent packets can be generated respectively: $x(\lambda a + \mu b), x \in \mathrm{GF}(2^n)\backslash 0$

Proof 1 *Given: Packets $a, b \in GF(2^n)$. Then there exist 2^{2n} packets in the form $\lambda a + \mu b$, $\lambda, \mu \in GF(2^n)$. By means of the zero vector $(0a+0b)$ there can be no generation of a linearly independent packet, thus $2^{2n}-1$ packets remain. Of each of these, 2^n-1 packets are linearly dependent upon one another. Thus $(2^{2n}-1)/(2^n-1) = 2^n+1$ linearly independent packets remain.* □

Up to now it has been assumed that in each case two packets are combined by a binary operation for forward error correction; however, it goes without saying that there can be any number of packets $m \geqslant 2$. The number of possible redundancy packets increases correspondingly. Unfortunately, the demand for pairs with linear independence no longer suffices. Rather all combinations of m packets must be linearly independent of one another.

Theorem 2 *In a field GF(2^n) consisting of t packets $a_1 \ldots a_t$ ($t \geqslant 1$) exactly $2^{n(t-1)} + 2^{n(t-2)} + \cdots + 2^n + 1$ pairs of linearly independent packets in the form $\lambda_1 a_1 + \cdots + \lambda_t a_t$ with $\lambda_i \in GF(2^n)$ can be generated.*

The proof follows the proof of theorem 1.

The following example demonstrates the meaning of the terms in the sum. We consider that "3 out of n" in the field GF(2^8). The sum in this case is: $2^{16} + 2^8 + 1 = 65793$. The set of representatives of the respective equivalence class is as follows: $\{(0,0,1),(0,1,i),(1,i,j)\}$ with $i,j \in GF(2^8)$. All elements of the form $(0,0,k)$ have $(0,0,1)$ as representative. Elements of the form $(0,k,l)$, $k > 1$ are multiples of $(0,1,i)$ and vectors of the form (k,l,m), $k > 1$ are multiples of an element of the form $(1,i,j)$.

Unfortunately, this statement is of little use because the packets used in reconstruction must all be mutually independent, not only just as pairs. That is, a set $\mathbb{M}$ of three elements must be found so that in each instance three elements are linearly independent. A formal description of the problem follows:

If $\mathbb{A}_n$ is the set of all sets containing three elements over GF(2^n):
$\mathbb{A}_n := \{(x,y,z) : x,y,z \in GF(2^n)\}$
then the set $\mathbb{M}_n$ is defined as follows:
$\mathbb{M}_n := \{a \in \mathbb{A}_n : \forall a,b,c : a \neq b \neq c \neq a, \det(a,b,c) \neq 0\}$
$\det(a,b,c)$ is the determinant of the matrix (a,b,c). If this determinant is not equal to zero, the three vectors are linearly independent. From here an algorithm can be easily derived for finding the set $\mathbb{M}_n$:

Algorithm 1
Algorithm for finding the set of linearly independent triples over the set GF(2^n):

1. $M_n := \{a,b\}, a,b \in (\mathrm{GF}(2^n))^3$, *$a,b$ linearly independent.*
2. *For all elements a out of* $(\mathrm{GF}(2^n))^3$*:*
   In the event that for all pairs of elements b,c $(b \neq c)$ out of M_n: $\det(a,b,c) \neq 0$
   then $M_n := M_n \cup \{a\}$

For GF(2^2) this algorithm can still be carried out by hand. The result can consist of the following vectors:
$\mathbb{M}_2 = \{(0,0,1),(0,1,0),(1,0,0),(1,1,1),(1,\alpha,1+\alpha),(1,1+\alpha,\alpha)\}$.
If the original set $\mathbb{A}_2$ is sorted differently, the set $\mathbb{M}_2$ contains other elements; their cardinality, however, remains the same (without proof). Using algorithm 1 ten elements were found for GF(2^4), and 66 elements for GF(2^8).

The algorithm can be drastically accelerated with the aid of theorem 2: Instead of taking the elements a in step 2 of algorithm 1 out of the set GF(2^n)^3, these can be taken from

the set of their representatives. For $GF(2^8)$ this set has only 65 793 elements as opposed to over 16777216 elements in $GF(2^8)^3$.

Let us now consider the corrective power and the overhead of our scheme. The more linearly independent packets we include in the transmission, the higher the corrective power (i. e. chance of restauration), but the higher also the overhead. For example, only minimal protection is provided by a single additional packet for every ten data packets. This results in a redundancy of only 10 percent. The disadvantage is that in the event of a lost packet the recipient must have all of the other ten packets at hand before the lost packet can be restored. This would lead to an increased delay in transmission. But with our scheme it is now easy to generate a larger number of redundant packets, thereby strongly increasing the reconstruction probability. For example, using a code out of the field $GF(2^8)$, 257 pairs of linearly independent packets could be generated out of only two data packets. This goes to the other extreme in increasing the data rate, but providing much better correction probability.

As we have seen, the method described enables the generation of an *adaptable redundancy*. Using the same method of calculation at the sender and recipient, the different segments of the data stream can be protected with varying degrees of correction probability and overhead. Therefore we call our method AdFEC (Adaptable Forward Error Correction).

4 Error Correction Probabilities

In this section we analyze the quality of the AdFEC method. We compare AdFEC with two alternatives: duplicating each packet, or no redundancy at all (i. e. no error correction).

Let n be the number of packets to be send and m be the number of packets that are added for error correction. Furthermore, let q be the probability that a packet arrives (i. e. $1 - q$ is the probability of loosing a particular packet). We assume that the loss of packets is an independent process. We use the following notation to denote the probabilities that the original n packets can be recovered:

$p(n, m)$	under the assumption that m additional packets are added;
$p_d(n)$	under the assumption that every packet is send twice (duplication);
$p_n(n)$	no forward error correction.

To obtain a formula for $p(n, m)$ note that the probability that exactly i out of the $n + m$ packets get lost is $q^{n+m-i}\,(1-q)^i \binom{n+m}{i}$. In order to reconstruct the original n packets

at most m packet losses are tolerable. Taking the sum over all possible cases we get

$$p(n,m) = \sum_{i=0}^{m} q^{n+m-i} (1-q)^i \binom{n+m}{i}$$

For the analysis of $p_d(n)$ note that the original n packets can be reconstructed only in the case of least one of the two duplicates arrives. Thus, if i packets are lost, the original n packets arrived at the receiver if the lost packets were all different. There are $2^i \binom{n}{i}$ different cases for loosing exactly i different packets. This yields

$$p_d(n) = \sum_{i=0}^{n} q^{2n-i} (1-q)^i 2^i \binom{n}{i}$$

Clearly for the function $p_n(n)$ we have

$$p_n(n) = q^n$$

The overhead in the case of sending every packet twice is 100%. In the case where m additional packets are send the overhead is 100m/n%.

The following tables characterize the situation for $q = 0.9$ and $q = 0.7$. In the first case we have $p_n(10) = 0.35$ and $p_d(10) = 0.90$ and in the second case we have $p_n(10) = 0.03$ and $p_d(10) = 0.39$.

Probability $q = 0.9$

numb. of add. packets	$p(10,m)$	overhead in %
1	0.697	10
2	0.889	20
3	0.966	30
4	0.991	40
5	0.998	50

Probability $q = 0.7$

numb. of add. packets	$p(10,m)$	overhead in %
1	0.113	10
2	0.253	20
3	0.421	30
4	0.584	40
5	0.722	50

Thus, in the case $q = 0.9$ with an overhead of only 20%, nearly the same arrival probability can be achieved by our algorithm as in the case where every packet is sent twice (i. e. the overhead is 100%). In the case $q = 0.7$ the situation is similar: with an overhead of only 30% a better result can be achieved than with duplication. Moreover, the arrival probability with only 10% overhead is 69% (11%) compared to 35% (3%) without error correction. Hence, our adaptive method is superior to the simple method of duplicating the packets, and with very little overhead it is possible to improve the arrival probability by a factor of 5 - 10 over no error correction.

5 Experience with Adaptable Forward Error Correction for a Digital Movie System

We have implemented the AdFEC algorithms described above in the context of a digital movie system. We decided to use 0, 1 or 2 redundant packets for two packets, depending on the importance of the segments transmitted. We use the operations $\circ$ and $\bullet$ introduced in Section 3.1.

5.1 Structure of the AdFEC Data Stream

In order to carry out the correction of errors, it is necessary to put the segments of the data stream to be protected and the redundant segments into different packets. The code described in the preceding section can now, for example, be employed as follows:

- Priority 3 ("loss does no harm"): packets are transmitted without redundancy.
- Priority 2 ("small amount of loss acceptable"): packets are transmitted in such a manner that for every two data packets A and B a third AdFEC packet A$\circ$B is generated.
- Priority 1 ("loss not acceptable"): packets are transmitted in such a manner that for every two data packets A and B, two AdFEC packets A$\circ$B and A$\bullet$B are generated and transmitted.

5.2 Implementation and Performance

AdFEC was implemented within the Movie Transmission Protocol (MTP) [8, 3] in the framework of the XMovie-Project [7] at the University of Mannheim. It protects the color lookup table transmission; damage to or a loss of the color lookup table within a movie data stream would have catastrophic results. In our implementation the field $GF(2^2)$ was used. Since the size of the color lookup table is only about 1 kbyte per image, compared to approx. 50 kbytes of pixel data, and a high degree of security is required, the error protection (2 out of 4) was chosen for the color lookup table packets. Because reconstruction requires only XOR and multiplication by α, AdFEC could be implemented with very few machine instructions. The multiplication uses a small table in memory; table lookup is more efficient than explicit computation at runtime, and can be carried out in just a few machine instructions. Since the addition corresponds to the XOR operation

which is carried out in hardware, the total efficiency of AdFEC is very high. AdFEC was written in C on UNIX workstations (SUN-10 and DEC-5000). Because Standard C was used exclusively, porting of the error correction procedure onto other architectures is very easy.

6 Summary

With the use of forward error correction several problems arising from the use of traditional error protection for multimedia data streams can be solved. The principal advantages are the maintenance of isochronous data flow, and ease of use in multicast environments.

This paper started with defining the requirements of an error correction method for isochronous data streams. It became apparent that only Forward Error Correction methods could be considered since retransmission destroys the timing relation between sender and receiver. The error correction method must be capable of correcting the loss of entire packets. Moreover, the same algorithm should be able to work with adjustable redundancy.

For this reason a central part of this paper concentrate on the generation of redundant information for FEC. Polynomial algebra represents a good source of adequate fields with suitable operations according to the expected rates of loss and the required error protection. It turned out, though, that the generation of several linearly independent packets is not trivial at all. We have presented an algorithm for this purpose, and investigated its error correction power and degree of redundancy.

The algorithm requires only very little computation, and therefore can be implemented very efficiently. An implementation in hardware directly on the network adapter card is conceivable. AdFEC was successfully implemented within the XMovie-Project.

References

[1] E.W. Biersack. Performance Evaluation of Forward Error Correction in ATM Networks. *Computer Communication Review*, 22(4):248–257, October 1992.

[2] E.W. Biersack. Performance Evaluation of Forward Error Correction in an ATM Environment. *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 4 (1993), pp. 631-640

[3] O. Böhrer. Fehlerkorrektur und Flußkontrolle für multimediale Datenströme. Master's thesis, Lehrstuhl für Praktische Informatik IV, Universität Mannheim, 1993.

[4] B. Hornfeck. *Algebra.* de Gruyter, Berlin, 1976.

[5] S. Kaneda and E. Fujiwara. Single Bit Error Correction Double Bit Error Detecting Codes for Memory Systems. *IEEE Transactions on Computers*, C-31(7):596–602, July 1982.

[6] K. Kawashima and H. Saito. Teletraffic Issues in ATM Networks. *Computer Networks and ISDN Systems*, 20:369–375, 1990.

[7] B. Lamparter and W. Effelsberg. X-MOVIE: Transmission and Presentation of Digital Movies under X. In R.G.Herrtwich, editor, *2nd International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, November 1991*, volume 614 of *Lecture Notes in Computer Science*, pages 328–339. Springer-Verlag Berlin Heidelberg, 1992.

[8] B. Lamparter, W. Effelsberg, and N. Michl. MTP: A Movie Transmission Protocol for Multimedia Applications. *ACM Computer Communications Review*, 22:71–72, July 1992.

[9] S. Lin and J. Costello. *Error Control Coding.* Prentice Hall, 1983.

[10] A.J. McAuley. Reliable broadband communication using a burst erasure correcting code. *Proc. ACM SIGCOMM '90 Symposium. Communications Architectures and Protocols*, ACM Computer Communication Review, Sept. 1990, vol.20, No.4, pp.297-306.

[11] H. Ohta and T. Kitami. A Cell Loss Recovery Method Using FEC in ATM Networks. *IEEE Journal on Selected Areas in Communications*, 9(9):1471–1483, December 1991.

[12] N. Shacham and P. McKenny. Packet recovery in High-Speed Networks using Coding. In *Proc. INFOCOM 90, San Francisco*, June 1990.

[13] L.B. Vries and K. Odaka. Circ — The Error Correcting Code for the Compact Disk. In *The AES Premier Conference — The New World of Digital Audio*, June 1982.

[14] L. Zhang and K.W. Sarkies. Modelling of a Virtual Path and its Application for Forward Error Recovery Coding Schemes in ATM Networks. In *Proc. SIGCOM'91 Singapore*, September 1981.