誤り制御符号 (2)：CRC 符号

1　目的

CRC 符号の生成と検査を行うプログラムの作成を通じて，誤り制御符号についての理解を深める．

2　ネットワークにおける転送メッセージと誤り制御符号

誤り制御を実現するため，ネットワーク上で交換されるデータ（**符号語**）には，本来のデータ（**情報語**）に加えて，検査用の符号（**検査語**）が含まれている．このような符号語を**誤り制御符号（Error Control Code）**と呼ぶ．受信局は，制御符号と予め定められた検出アルゴリズムを使用して，誤りの有無を判定する．図 1 は，TCP/IP プロトコルのトランスポート層で交換されるデータ（TCP セグメント）の形式を簡略化したものである．ヘッダ部に含まれている**チェックサム（Check Sum）**が検査語に該当する．送信局はヘッダの一部とデータ部の内容に基づいて，チェックサムの内容を生成する．受信局は，受信したセグメントの内容からチェックサムの値を求め，受信したセグメント内のそれと比較することにより，エラーの有無を判断する．

また，LAN（Local Area Network）の物理層とデータリンク層の規格として最も普及しているイーサネット（Ethernet）では，図 2 に示す形式のデータ（データリンク層ではフレームと呼ぶ）を使用している．フレームの最後尾に**フレームチェックシーケンス（Frame Check Sequence; FCS）**と呼ばれる検査語がある．送信局はヘッダ部とデータ部を合わせた 60〜1,514 バイトのデータをもとに，フレームチェックシーケンスを生成する．イーサネットでは，フレームチェックシーケンスを生成するアルゴリズムとして，**CRC32（Cyclic Redundancy Checks 32）**が使用されており，これは今回の実験で学習する CRC 符号の一種である．イーサネットを採用している多くの LAN では，上位プロトコルとして TCP/IP を用いている．つまり，イーサネットフレームのデータ部には TCP セグメントが含まれている．従って，多くの LAN では誤りの検査が二重に施されている．さらに、電子メールでは，アプリケーション層においてパリティ符号を用いた誤り制御を実施しているため，三重の検査が施されている．

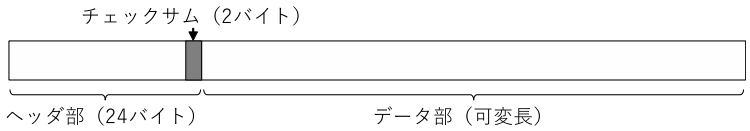


図 1　TCP セグメントとチェックサム

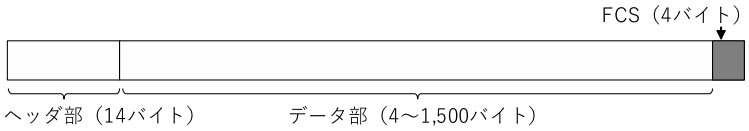


図 2　イーサネットフレームとフレームチェックシーケンス

3　CRC 符号

CRC 符号（Cyclic Redundancy Check Code）は**巡回符号（Cyclic Code）**と**ハミング符号（Hamming Code）**の性質を併せ持つ誤り検出符号で，IEEE 802.3（イーサネットの標準規格）や IEEE 802.11（無線 LAN の標準規格）のフレームチェックシーケンス（FCS）として採用されている．

まず，FCS の基本概念について学習する．図 3 に，FCS の基本概念を示す．情報語をある符号で除算し，その剰余を FCS の値とする．受信局は，受信した符号語を送信局と同じ符号で除算し，その剰余を FCS の値と比較することにより，誤りの有無を判定する．FCS を用いた誤り検出では，データを除算する符号を適切に選択することが重要である．特に，イーサネットのように多様で可変長のデータが交換されるネットワークにおいて誤り検出を行うためには，次の条件を満たす必要がある．

1. ビット数が等しく内容が異なるデータに対しては，異なる剰余が算出される．
2. ひとつのデータ内では，どの桁から除算を開始しても同じ剰余が算出される．
3. 符号語の中で，情報語と検査語とが独立・分離している．

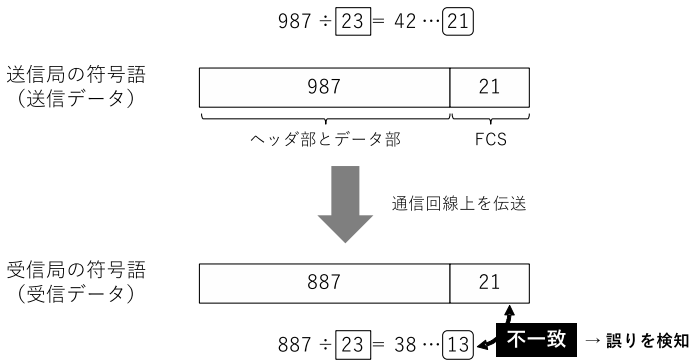


図 3　フレームチェックシーケンスの基礎

これらの条件を満たすため，巡回符号を利用した誤り検出符号が考案された．

続いて、巡回符号について学習する．いま，k ビットの情報語を（$f_{k-1}f_{k-2}\cdots f_1f_0$）とする．また，$m+1$ ビット（$m \leqq k$）の誤り検出符号を（$g_mg_{m-1}\cdots g_1g_0$）とする．これらのビット列を下記のような多項式で表現する．$F(x)$ を情報多項式，$G(x)$ を生成多項式と呼ぶ．なお，以降の多項式に関する演算はすべて Modulo2 演算である（付録 A 参照）．

$$F(x) = f_{k-1}x^{k-1} + f_{k-2}x^{k-2} + \cdots + f_1x + f_0$$
$$G(x) = g_mx^m + g_{m-1}x^{m-1} + \cdots + g_1x + g_0$$

ここで，$G(x)$ は $x^{k+m}-1$ の素因数のうち，最大次数が m となるものを採用する．すると，$x^mF(x)$ を $G(x)$ で除算した際の剰余 $R(x)$ は最大で m ビットとなり，多項式表現した際の最大次数は $m-1$ となる．これを用いて，符号多項式 $P(x)$ を次のように示す．

$$P(x) = x^mF(x) + R(x)$$

これは，情報語を m ビットだけ左シフトし，m ビット分の下位桁（左シフト後はすべて 0 になっている）に検査語（剰余）をあてはめたものになる．剰余 $R(x)$ が FCS に該当するため，図 3 及び図 4 に示したように，送信局は情報語に続けて検査語を送信する．

ここで，$P(x)$ を任意の n ビット（$n \geq 0$）だけ巡回シフトした際の多項式はいずれも $G(x)$ で割り切れる（$P(x)$ も $G(x)$ で割り切れる）．このことは，巡回シフトしたものも（別の情報語の）符号語であることを意味している．このような性質を持つ符号を巡回符号と呼ぶ．

誤り検出符号 (k=4, m=3)	1011
情報語	1010
生成多項式	$G(x) = x^3 + x + 1$
情報多項式	$F(x) = x^3 + x$
剰余（検査語）	$R(x) = x^mF(x) \div G(x) = x + 1$ （2進数表現では011）
符号語（P(x)に相当）	1 0 1 0 0 1 1
1ビット巡回シフト後	0 1 0 0 1 1 1
2ビット巡回シフト後	1 0 0 1 1 1 0
6ビット巡回シフト後	1 1 0 1 0 0 1

図 4　CRC 符号による巡回符号の生成

送信局は，図 4 に示したように送信データを生成し，通信回線上に送出する（剰余 $R(x)$ の計算過程は付録 B を参照）．一方，受信局は受信データの先頭から検査を開始し，1 ビット受信するごとに，最近受信した $k+m$ ビットのデータを $P(x)$ とみなし，これを巡回シフトしながら $G(x)$ で割り切れるか否かを調査する．受信データ内に誤りがない場合，データ部の最終ビットに至るまでのすべてのビット列の組合せが $G(x)$ で割り切れる．

また，CRC 符号は誤り訂正符号としても使用できる．ここで，情報語のビット数を k，検査語のビット数を m とするとき，次式を満たすようにこれらの値を決定する．

$$2^m - m \geq k + 1$$

このとき，符号語及びこれを巡回シフトして生成されるビット列は，いずれもハミング符号の性質を持つ．つまり，1 ビットの誤りが発生した場合，誤りを含む符号語を $G(x)$ で除算したときの剰余はシンドロームとなり，誤りが発生した位置と対応付けることができる．実用化されている CRC 符号では，上式の左辺と右辺が等しくなるような m と k を使用している．なお，複数ビットの誤りについては，巡回符号の性質によりこれを検出／訂正することは可能であるが，正確に検出／訂正できないケースも存在する．

CRC 符号は、可変長のデータに対して誤り検出を行う手法の中では、誤り検出能力が高く，回路の構成が比較的簡易であるため，さまざまなネットワークや伝送路に採用されている．符号化／復号化の回路はいずれも，$m+1$ 個のシフトレジスタと $m+1$ 個の排他的論理和によって構成できる．符号化／復号化の回路への入力はいずれも 1 ビットである．ただし，符号語は誤り検出符号よりもビット数が大きいため，一度に符号語の全体を符号化／復号化することはできない．このため，上位桁の剰余を下位桁にフィードバックする回路も必要になる．

4 CRC 符号の実験

4.1 プログラムの作成

次のように入力して，CRC 符号のソース・プログラムを実験用ディレクトリにコピーする．

```
$ cd ~/cel-B
$ cp sample-programs/crc.c .
```

コピーしたソース・プログラムには，各自で追加すべき部分がいくつかある．テキストエディタ等を使用して，必要な部分を追加する．

ソース・プログラムが完成したら，次のように入力してコンパイルを行う．コンパイルが正しく終了すると，`crc` という実行可能プログラムが生成される．

```
$ gcc -o crc crc.c
```

4.2 プログラムの実行

次のように入力して，CRC 符号のプログラムを実行する．

```
$ ./crc
```

4.3 CRC 符号の考察

作成したプログラムを何度か実行し，CRC 符号の性質について考察する．1 ビットの誤りを訂正できることも確認する．

5 報告内容

レポートには，以下の内容を記載すること．

5.1 実行結果

次節の考察を行う材料となる結果を掲載すること．

5.2 実行結果に対する考察

前節に掲載した実行結果を用いて，1 ビットの誤りを正しく訂正できていることを示すこと．

5.3 課題

(1) 生成多項式 $G(x) = x^3 + x + 1$ を用いて 4 ビットの情報語 $(0101)_2$ から 7 ビットの符号語を生成するとき、符号語が $G(x)$ で割り切れることを示せ．また，符号語のどこか 1 ビットを変更した語を作成し，これが $G(x)$ で割り切れないことを示せ．

6 追加提出物

レポートに加え，修正したプログラムのソースコード (`crc.c`) を成果物提出先ディレクトリに提出すること．

参考文献

[1] 福永邦雄，泉正夫，荻原昭夫，「コンピュータ通信とネットワーク 第 5 版」，共立出版，2002 年．

[2] 荻原春生，中川健治，「情報通信理論 1 ―符号理論・待ち行列理論―」，森北出版，1997 年．

[3] イエルン・ヨステンセン，トム・ホーホルト，「誤り訂正符号入門」，森北出版，2005 年．

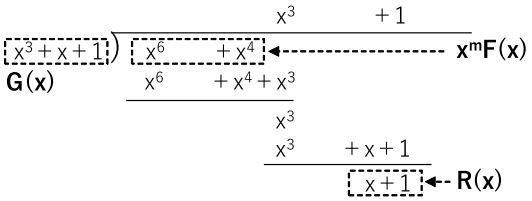
付録 A　Modulo2 演算

Modulo2 演算は，2 進数の各ビットに対して次のような規則を適用する演算である．

$$
\begin{array}{lll}
0+0=0 & 0\times 0=0 & 0-0=0 \\
0+1=1 & 0\times 1=0 & 0-1=1 \\
1+0=1 & 1\times 0=0 & 1-0=1 \\
1+1=0 & 1\times 1=1 & 1-1=0
\end{array}
$$

付録 B　Modulo2 演算に基づいた多項式の除算

まず、テキストの本文中の図 4 にある $x^m F(x)$ と $G(x)$ の除算の過程を以下に示す．



従って、$P(x) = x^m F(x) + R(x) = x^6 + x^4 + x + 1$ となる．
また，同様の除算により，$P(x)$ が $G(x)$ で割り切れることが分かる．