smartlearn
virtual learning. real success.

MODUL 114

TEIL 5: FEHLER IN DER DATENÜBERTRAGUNG

Daniel Schär

Aufgaben

Aufgabe 5.1: Codes mit erhöhter Redundanz

Neben dem bereits bekannten **1 aus 10 Code** gibt es auch den **2 aus 5 Code** um die Übertragungsqualität zu erhöhen. Dabei werden pro Dezimalziffer jeweils 5 Bit verwendet wovon zwei den Wert 1 haben und die restlichen drei auf 0 gesetzt sind.

Geben Sie für beide Codes die folgenden Werte an:

a) Hamming-Abstand
b) Anzahl möglicher Kombinationen
c) Anzahl gültiger und ungültiger Kombinationen
d) Redundanz

Aufgabe 5.2: Paritätsbit

Folgende Datenblöcke (jeweils 1 Byte) wurden mit angehängtem Paritätsbit (Even Parity) übertragen. Welche Blöcke sind fehlerfrei angekommen?

a) 110011001
b) 111001111
c) 000000111
d) 000000000
e) 110000101
f) 001010110

Aufgabe 5.3: Paritätsbit pro Zeile und pro Spalte

Wir erweitern die Idee des Paritätsbits ein wenig, indem wir nach 8 Bytes (Zeilen) jeweils ein Paritätsbit pro Spalte berechnen und dies dem Empfänger auch zusenden.

									Paritätsbit Zeile
Byte 1	1	1	0	0	0	1	1	1	**1**
Byte 2	0	0	0	1	1	1	0	1	**0**
Byte 3	1	1	1	1	1	1	1	0	**1**
Byte 4	0	0	0	0	0	1	1	1	**1**
Byte 5	0	1	1	0	0	1	1	0	**0**
Byte 6	1	0	0	1	0	0	1	0	**1**
Byte 7	1	0	1	0	1	0	0	0	**0**
Byte 8	0	1	0	1	0	1	1	0	**0**
Paritätsbit Spalte	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	

a) Es wurde ein Bit falsch übertragen. Finden und korrigieren Sie es.

b) Berechnen Sie die Redundanz, welche die Paritätsbits verursachen.

Aufgabe 5.4: Hamming Code

Ein mit Hamming-Code gesichertes Byte wurde wie folgt (fehlerhaft) von einem Empfänger entgegengenommen:

1001 0110 0101

a) An welcher Stelle ist der Fehler aufgetreten?
b) Wie lautete das übertragene Byte im Original?
c) Wie gross ist die durch den Hamming-Code verursachte Redundanz?

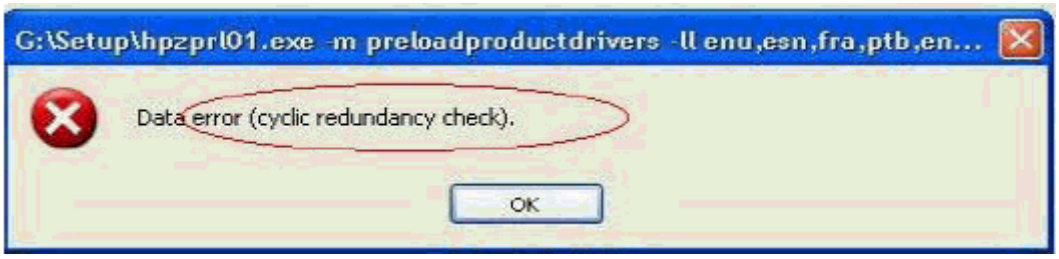
Zusatzaufgabe für Interessierte: CRC-Prüfung

Lesen Sie das Dokument „Funktionsweise CRC-Prüfung“ im Anhang zu diesem Arbeitsblatt und spielen Sie das Beispiel mit eigenen m = 1010101 und g = 1001 durch.

Anhang: Die CRC-Prüfung (Zyklische Redundanzprüfung)

Der vorliegende Artikel enthält Angaben der Webseiten de.wilkipedia.org und www.kryptographiespielplatz.de.

Die zyklische Redundanzprüfung (englisch cyclic redundancy check, daher meist CRC) ist ein Verfahren zur Bestimmung eines Prüfwerts für Daten, um Fehler bei der Übertragung oder Speicherung erkennen zu können. Im Idealfall kann das Verfahren sogar die empfangenen Daten selbständig korrigieren, um eine erneute Übertragung zu vermeiden.



In der Variante CRC-32 (sehr bekannte Variante, implementiert im Ethernet-Standard 802.3 sowie in den Dateiformaten PNG und ZIP) hat das (fixe) Generator-Polynom eine Grösse von 32 Bit. Es lautet 0x04C11DB7

Im folgenden Beispiel verwenden wir aber aus Gründen der Nachvollziehbarkeit nur sehr kleine Zahlen.

1. Prüfziffer ermitteln

Als erstes benötigen wir ein Generator-Polynom *g* mit *n* Stellen. Es wird durch den jeweiligen Standard fix vorgegeben. Unser Generatorpolynom sei *g* = 1101.

Dann brauchen wir natürlich eine zu übermittelnde Nachricht *m*.
Unsere Nachricht sei *m* = 1001010.

In **einem** ersten Schritt erhält die Nachricht einen Anhang aus *n-1* Nullen. Also eine Null weniger, als das Generatorpolynom Stellen hat.
Somit ergibt sich *m'* = 1001010000

Die Nachricht *m'* wird nun durch das Generatorpolynom „dividiert". Der dabei entstehende Rest ist dann die eigentliche Prüfsumme. Die Operation heisst Polynom-Division und geht wie folgt:

- Nachricht aufschreiben, Generator-Polynom linksbündig darunter
- XOR-Verknüpfung berechnen (unter dem Strich)
- Nächste Stelle „herunter" holen
- Generator-Polynom mit seiner ersten 1 unter die erste 1 schreiben
- Wiederum XOR-Operation, usw. bis alle Stellen „geholt" sind.

```
1 0 0 1 0 1 0 0 0 0
1 1 0 1
---------
0 1 0 0 0
  1 1 0 1
  -------
  0 1 0 1 1
    1 1 0 1
    -------
    0 1 1 0 0
      1 1 0 1
      -------
      0 0 0 1 0 0 0
          1 1 0 1
          -------
          0 1 0 1      = Prüfziffer
```

2. Nachricht übermitteln

Nun wird die Prüfsumme zur Nachricht *m'* addiert. Das ergibt den zu übermittelnden Rahmen *m''*:

m'' = 1001010000 + 101 = 1001010101

3. Nachricht nach dem Empfang überprüfen

Der Empfänger arbeitet mit demselben CRC-Standard. Er kennt also das Generator-Polynom und weiss, dass die letzten drei Stellen eine Prüfziffer sind und nicht zur eigentlichen Nachricht gehören.

Nach Erhalt des Rahmens *m''* führt der Empfänger seinerseits die Polynomdivision durch. Falls diese ohne Rest aufgeht, kann davon ausgegangen werden, dass die Übermittlung fehlerfrei verlaufen ist.

```
1 0 0 1 0 1 0 1 0 1
1 1 0 1
---------
0 1 0 0 0
  1 1 0 1
  -------
  0 1 0 1 1
    1 1 0 1
    -------
    0 1 1 0 0
      1 1 0 1
      -------
      0 0 0 1 1 0 1
          1 1 0 1
          -------
          0 0 0 0      Kein Rest
                       Übertragung korrekt
```

Nun können die letzten drei Stellen abgetrennt werden und es kann mit der Originalnachricht **m** weitergearbeitet werden.