Verteilte Systeme (Distributed Systems) Frühling-Sommer 2017

Aufgabe 1:

**Hamming-Codes**

**Gruppe:** Emanuela Giovanna Calabi (13186), Angelo Rosace (13386)

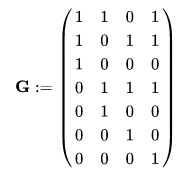
**Beschreibung:**

Die Aufgabe betrachtete die Implementierung des Hamming Code [7,4].

[7,4] bedeutet, dass vier data bits in 7 bits codiert werden. Während der Kodierung drei parity bits werden zu die vier data bits hinzugefügt. Sie werden nicht in zufüllige Stellen gestellt sondern in Stelle 1,2 und 4. Am ende des Prozess, die codierte daten sehen so aus: p1,p2,d1, p3,d2,d3,d4; wo p für „parity bit“ und d für „data bit“ steht.

Diese Art von Hamming Code kann am meisten ein Fehelr auf ein bit erkennen und korregieren.

Um den Hamming-Codes Kodierungsprozess zu implementieren haben wir zwei Matrizen verwendet. Die zwei Matrizen heißen Hammings Matrizen. Wir haben entschieden anstatt von 0 und 1, false und true aufgrund der Semplizität und der Effizienz des Codes zu benutzen.





**G** (Code-Generatormatrix) ist verwendet um die Kodierung durch zu führen.

Die 7 codierte bits sind das Ergebnis des Modulo 2 (boolesche XOR) des Produkt (boolesche AND) zwischen G und einen Vektor (**v**), der die 4 data bits enthält.

**H** ist benutzt um zu beweisen ob die Übertragung der Daten Fehlern generiert hat.

Das Ergebnis des Produkt zwischen **v** und **H** ist noch ein Vektor (**e**). Wenn dieser Vektor nur aus 0 (false) besteht bedeutet es, dass kein Fehler gibt. Wenn es ein oder mehr 1 (true) gibt bedeutet es, dass ein Fehler gibt.

**e** representiert als Binär-zahl die Stelle wo das Fehler liegt. Um das Fehler zu korregieren mussen wir das bit, dass in die von e representierte Stelle ligt umdrehen.

Von diesem Prozess erhalten wir ein 7 bit boolesche Vektor die noch dekodiert werden muss.

Um es zu dekodieren brauchen wir noch eine Matrix.



Wenn wir unseren fehlerfreien Vektor mit **R** multiplizieren erhalten wir ein 4 bit boolesche vektor.

Am Ende des Dekodierungsprozess mussen wir den booleschen Vektor in einer Binär-Zahl umwandeln. So erhalten wir die gleiche 4 Daten-bits die wir am Anfang hätten.