Systeme II

3. Die Datensicherungsschicht

Christian Schindelhauer
Technische Fakultät
Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg

Version 12.05.2016

Fehlererkennung: CRC

- Effiziente Fehlererkennung: Cyclic Redundancy Check (CRC)
- Praktisch häufig verwendeter Code
  - Hoher Fehlererkennungsrate
  - Effizient in Hardware umsetzbar
- Beruht auf Polynomarithmetik im Restklassenring Z_2
  - Zeichenketten sind Polynome
  - Bits sind Koeffizienten des Polynoms

Galois-körper

$\bar{3}+\bar{5} = \cancel{\bar{8}}\ \bar{8}$

$\bar{3}\cdot\bar{5} = \bar{15}$

$\frac{\bar{8}}{\bar{5}} = \frac{\bar{8}}{\bar{6}}$

→ endlichen körper GF

finite field $\mathbb{F}$

CoNe Freiburg

Rechnen in Z_2

$x + 1 = y$

$x = y - 1$

$1 = -1$

- Rechnen modulo 2:
- Regeln:
  - Addition modulo 2 = Xor = Subtraktion modulo 2
  - Multiplikation modulo 2 = And

$A - B$

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	$\boxed{0}$!

XOR

A	B	A - B = A+B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A * B
0	0	0
0	1	0
1	0	0
1	1	1

$\boxed{\text{AND}}$

- Beispiel: $0 + (\underbrace{1 * 0}_{0}) + 1 + (\underbrace{1 * 1}_{1}) = 0$

$A + B = \underline{A + B \bmod 2}$

$A * B = \underline{A \cdot B \bmod 2}$

CoNe Freiburg

Polynomarithmetik modulo 2

- Betrachte Polynome über den Restklassenring Z_2
  - $p(x) = a_n x^n + \ldots + a_1 x^1 + a_0$
  - Koeffizienten a_i und Variable x sind aus $\cancel{Z}$ {0,1}
  - Berechnung erfolgt modulo 2
- Addition, Subtraktion, Multiplikation, Division von Polynomen wie gehabt

Fehlererkennung: CRC

- Effiziente Fehlererkennung: Cyclic Redundancy Check (CRC)
- Praktisch häufig verwendeter Code
  - Hoher Fehlererkennungsrate
  - Effizient in Hardware umsetzbar
- Beruht auf Polynomarithmetik im Restklassenring Z_2
  - Zeichenketten sind Polynome
  - Bits sind Koeffizienten des Polynoms

x^4

1011

$x^2+1 = (x+1)^2 \quad | \quad 111$

$101 = 11 \cdot 11$

$$
\begin{array}{l}
10010101 : \underbrace{1011}_{g(x)} = 10101 \\
1011 \\
\hline
0010010\,1 \\
\;\;1011 \\
\hline
\;\;001001\,1 \\
\;\;\;\;\;1011 \\
\hline
\;\;\;\;\;\;\;10
\end{array}
$$

Rest $\underline{10}$

$\underline{\text{Mod}}$

Rechnen in Z_2

10111 +
00101
10010

Kein Carry

$x^4 + x^2 + x^1 + x^0$
$+ x^0 + x^2 =$
$\cancel{2 \cdot x^0} + x^1 + x^2$

- Rechnen modulo 2:
- Regeln:
  - Addition modulo 2 = Xor = Subtraktion modulo 2
  - Multiplikation modulo 2 = And

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A - B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A ¢ B
0	0	0
0	1	0
1	0	0
1	1	1

  - Beispiel: $0 + (1 \cdot 0) + 1 + (1 \cdot 1) =$

Polynomarithmetik modulo 2

- Betrachte Polynome über den Restklassenring Z_2
  - $p(x) = a_n x^n + \ldots + a_1 x^1 + a_0$
  - Koeffizienten a_i und Variable x sind aus 2 {0,1}
  - Berechnung erfolgt modulo 2
- Addition, Subtraktion, Multiplikation, Division von Polynomen wie gehabt

$$123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$$

$$123 \cdot 25$$

$$0 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0$$

$$010 \cdot 110$$

000
010
010

01100

Zeichenketten und Polynomarithmetik

- Idee:
  - Betrachte Bitstring der Länge n als Variablen eines Polynoms
- Bit string: $b_n b_{n-1} \ldots b_1 b_0$

  Polynom: $b_n x^n + \ldots + b_1 x^1 + b_0$
  - Bitstring mit (n+1) Bits entspricht Polynom des Grads n
- Beispiel
  - A xor B = A(x) + B(x)
  - Wenn man A um k Stellen nach links verschiebt, entspricht das
    - $B(x) = A(x)\, x^k$
- Mit diesem Isomorphismus kann man Bitstrings dividieren

Polynome zur Erzeugung von Redundanz: CRC

$x^4 \; x^3 \; x^0$

11001

$g(x)$

- Definiere ein Generatorpolynom *G(x)* von Grad *g*
  - Dem Empfänger und Sender bekannt
  - Wir erzeugen g redundante Bits
- Gegeben:
  - Frame (Nachricht) M, als Polynom M(x)
- Sender
  - Berechne den Rest der Division r(x) = x^g M(x) mod G(x)
  - Übertrage T(x) = x^g M(x) + r(x)
    - Beachte: x^g M(x) + r(x) ist ein Vielfaches von G(x)
- Empfänger
  - Empfängt m(x)
  - Berechnet den Rest: m(x) mod G(x)

$10111010\underbrace{0000}_{g} : 11001 =$

$\underbrace{0110}_{r(x)}$

$\overbrace{10111010}^{M}\overbrace{0110}^{r}$

= 0 !

CRC Übertragung und Empfang

- Keine Fehler:
  - T(x) wird korrekt empfangen
- Bitfehler: T(x) hat veränderte Bits
  - Äquivalent zur Addition eines Fehlerpolynoms E(x)
  - Beim Empfänger kommt T(x) + E(x) an
- Empfänger
  - Empfangen: m(x)
  - Berechnet Rest m(x) mod G(x)
  - Kein Fehler: m(x) = T(x),
    - dann ist der Rest 0
  - Bit errors: m(x) mod G(x) = (T(x)+ E(x)) mod G(x)

$$= \underbrace{T(x) \bmod G(x)}_{0} + \underbrace{E(x) \bmod G(x)}_{\text{Fehlerindikator}}$$

CRC – Überblick

Sender

Original Frame M(x)

Generator Polynom G(X)

$r(x) = x^g M(x) \bmod G(x)$

Sende $T(x) = x^g M(x) + r(x)$

Kanal

Addiert Fehlerpolynom E(x)

Kein Fehler:
$E(x) = 0$

Empfänger

Empfang: $m(x) = T(x) + E(x)$

Berechnet Rest $(T(x) + E(x)) \bmod G(x)$

Falls Rest = 0 : Kein Fehler

If Rest ≠ 0: Fehler!

Der Generator bestimmt die CRC-Eigenschaften

- Bit-Fehler werden nur übersehen, falls E(x) ein Vielfaches von G(x) ist
- Die Wahl von G(x) ist trickreich:
- Einzel-Bit-Fehler: $E(x) = x^i$ für Fehler an Position i
  - G(x) hat mindestens zwei Summenterme, dann ist E(x) kein Vielfaches von G(x) ist
- Zwei-Bit-Fehler: $E(x) = x^i + x^j = x^j (x^{i-j} + 1)$ für i>j
  - G(x) darf nicht $(x^k + 1)$ teilen für alle k bis zur maximalen Frame-Länge
- Ungerade Anzahl von Fehlern:
  - E(x) hat nicht (x+1) als Faktor
  - Gute Idee (?): Wähle (x+1) als Faktor von G(x)
    - Dann ist E(x) kein Vielfaches von G(x)
- Bei guter Wahl von G(x):
  - kann jede Folge von r Fehlern erfolgreich erkannt werden
- Häufig:
  - G(x) wird als irreduzibles Polynom gewählt, dass heißt es ist kein Vielfache eines anderen (kleineren) Polynoms

CRC in der Praxis

$\frac{1}{2^{32}}$

- Verwendetes irreduzibles Polynom gemäß IEEE 802:
  - $x^{32} + x^{23} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Achtung:
  - Fehler sind immer noch möglich
  - Insbesondere wenn der Bitfehler ein Vielfaches von G(x) ist.
- Implementation:
  - Für jedes Polynom x^i wird $r(x,i)= x^i \bmod G(x)$ berechnet
  - Ergebnis von B(x) mod G(x) ergibt sich aus
  - $b_0\, r(x,0) + b_1\, r(x,1) + b_2\, r(x,2) + \ldots + b_{k-1}\, r(x,k-1)$
  - Einfache Xor-Operation

← 1011101100
1011