TUGAS TELEMATIKA

Oleh: Indra Oktavian

NIM : 1714321009

TEKNIK INFORMATIKA

UNIVERSITAS BHAYANGKARA SURABAYA

2019

Cyclic Redundancy Check (CRC)

Cyclic Redundancy check adalah pengecekan kesalahan yang banyak digunakan dalam sistem komunikasi data dan sistem transmisi data serial lainnya. CRC didasarkan pada manipulasi polinomial menggunakan modulo aritmatika. Beberapa CRC standar yang umum digunakan adalah CRC-8, CRC-12, CRC-16, CRC-32, dan CRC-CCIT.

Tujuan utama dari membuat sebuah algoritma pendeteksi error adalah untuk memaksimalkan kemungkinan mendeteksi error dengan menggunakan jumlah bit yang tidak berlebihan. Cyclic Redudancy Check menggunakan perhitungan matematik yang kompleks untuk mencapai tujuan ini. Sebagai contoh, 32bit CRC memberikan perlindungan terhadap terjadinya bit error dengan cara menampilkan pesan-pesan kesalahan sepanjang bit yang ada. Teori yang mendasari dari adanya CRC adalah teori matematis yang biasa disebut dengan finite fields.

Sebagai permulaan, asumsikan pesan yang memiliki (n+1)-bit, dimana n mewakili jumlah polynomial yang dimiliki, dengan notasi x^n. Nilai dari setiap bit dalam pesan tersebut digunakan sebagai koefisien untuk setiap notasi di dalam polynomial. Sebagai contoh, pesan 8bit yaitu 10011010 sesuai dengan polynomial:

$M(x) = 1.x^7 + 0.x^6 + 0.x^5 + 1.x^4 + 1.x^3 + 0.x^2 + 1.x^1 + 0.x^0 = x^7 + x^4 + x^3 + x^1$

Dapat kita asumsikan bahwa pengirim dan penerima bertukar nilai polynomial satu sama lain.

Untuk mencapai tujuan dari perhitungan CRC, penerima dan pengirim dari sebuah pesan itu harus memiliki kesepakatan yang sama dalam polynomial pembagi, C(x). k merupakan polynomial tertinggi dari C(x). Sebagai contoh, asumsikan bahwan $C(x) = x^3 + x^2 + 1$. Dalam kasus ini, k = 3. Sebagai contoh lain Ethernet pada dasarnya menggunakan polynomial dengan 32 derajat.

Ketika sebuah pengirim akan mengirimkan pesan M(x) dengan panjang n+1 bit, dimana pada kenyataannya bit yang dikirimkan adalah n+1 ditambah dengan k bit. Pesan yang sudah lengkap terkirim, termasuk dengan bit yang berlebihan, biasanya dilambangkan dengan P(x).

Apa yang akan kita lakukan selanjutnya adalah menyusun polynomial P(x) sedemikian rupa agar dapat tepat habis dibagi dengan C(x). Jika P(x) terkirimkan melalui sebuah jaringan, dan tidak ada error yang terdeteksi selama proses pengiriman, maka penerima seharusnya bisa membagi P(x) dengan C(x), tanpa ada sisa. Jika terjadi error di P(x) dalam proses pengiriman, maka polynomial yang diterima tidak akan habis dibagi dengan C(x), dan sisa yang ada tidak akan 0, yang mengartikan bahwa telah terjadi kesalahan.

Dalam topik bahasan ini kita bergelut dengan tipe khusus dari polynomial aritmetik, dimana koefisien hanya antara 1 atau 0, dan operasi yang digunakan adalah modulo 2 aritmetik. Karena yang dibahas adalah mengenai jaringan, bukan mengenai matematik, berikut merupakan kata kunci dari tipe aritmetik yang dibahas :

- Untuk semua polynomial B(x) dapat dibagi dengan polynomial pembagi C(x), hanya jika B(x) memiliki degree yang lebih tinggi daripada C(x)
- Untuk semua polynomial B(x) dapat dibagi sekali dengan polynomial pembagi C(x), hanya jika B(x) memiliki degree yang sama dengan C(x)
- Akan ada sisa, jika B(x) dibagi dengan C(x) dengan menggunakan operasi exclusive OR(XOR) untuk setiap pasang pesan yang memiliki koefisien sama

Sebagai contoh, polynomial $x^3 + 1$ dapat dibagi dengan $x^3 + x^2 + 1$, karena sama-sama memiliki degree sebanyak 3, dan sisa dari pembagian tersebut akan menghasilkan $0.x^3 + 1.x^2 + 0.x^1 + 0.x^0 = x^2$ (di dapat dengan cara meng-XOR-kan setiap komponen yang memiliki koefisien sama). Dalam istilah lain, dapat dikatakan bahwa 1001 dapat dibagi dengan 1101 dengan sisa 0100. Kita dapat melihat bahwa sisa yang didapat dari pembagian 1001 dengan 1101 merupakan XOR dari pesan dan generator.

Secara garis besar Cyclic Redundancy Check:

- Berguna untuk meminimalkan jumlah dari kelebihan bit dan memaksimalkan proteksi
- Jika diketahui ada bit string 110001, dapat kita asosiasikan polynomial terhadap setiap variable, sebagai berikut :
  $1.x^5 + 1.x^4 + 0.x^3 + 0.x^2 + 1.x^0 = x^5 + x^4 + 1$
  dengan derajat sebanyak 5. Susunan a-bit memiliki degree maksimum sebanyak a-1.
- Asumsikan M(x) merupakan pesan polynomial dan C(x) merupakan polynomial generator

- M(x) harus tepat habis dibagi dengan C(x)
- Ketika pesan M(x) dikirim dan M'(x) diterima, kita mempunyai M'(x) = M(x) + E(x)
- Penerima menghitung M'(x) dibagi dengan C(x), dan jika sisa dari pembagian tersebut bukan 0, maka telah terjadi kesalahan dalam proses pengiriman pesan
- Satu-satunya hal yang harus diketahui oleh penerima dan pengirim pesan adalah C(x)

Cara menghitung Cyclic Redundancy Check :

- Jika M(x) menjadi frame dengan *m* bits dan jika hanya sedikit polynomial generator dari *m* bits dikatakan sama dengan *r*.
- Jika *r* menjadi derajat dari C(x). Menambahkkan sebanya *r* bit 0 ke LSB dari frame, jadi frame berisi m+r bits dan sesuai dengan polynomial X^r M(x)
- Membagi bit string sesuai X^r M(x), bit string ini sesuai dengan C(x) menggunakan pembagian modulo 2
- Kurangi sisanya (yang bitnya selalu lebih sedikit dari r) dari string yang sesuai dengan X^r M(x) menggunakan pengurangan modulo 2 (penambahan dan pengurangan yang sama dalam modulo 2).
- Hasilnya adalah frame checksum yang akan dikirim. Disebut polinomial M '(x).

Untuk lebih jelasnya, kami memberikan contoh crc dari pengiriman data menggunakan CRC-8 dan CRC-CCIT sebagai berikut :

Terdapat pesan berisi "kelompok06" yang akan dikirimkan, maka :

Original Message = M(x) = kelompok06 = 01101011 01100101 01101100 01101111 01101101 01110000 01101111 01101011 00110000 00110110 (ASCII to binary)

1. CRC-8

Generator = C(x) = $x^8 + x^2 + x^1 + 1$ = 100000111

Message setelah ditambah bit 0 sebanyak 8 =

0110101101100101011011000110111101101101110000011011110110101100110000001101 1000000000

proses perhitungan ccrt-8(buka dengan notepad++)

2. CRC-CCIT

Generator = C(x) = $x^{16} + x^{12} + x^5 + 1$ = 10001000000100001

Message setelah ditambah bit 0 sebanyak 16 =

0110101101100101011011000110111101101101110000011011110110101100110000001101100000000000000000