ERROR DETECTION DAN CORRECTION PADA PENGIRIMAN DATA MENGGUNAKAN METODE HAMMING CODE DAN PARITY CHECK

Andi Gita Novianti[1)]
andigitaisme@gmail.com

Mursid[2)]
mursidjpr73@gmail.com

Fifin Kurniawati Nurdiaheni
fifin-kaen@gmail.com

[1, 2, 3)] **Program Studi Teknik Informatika, Fakultas Ilmu Komputer dan Manajemen**
Universitas Sains dan Teknologi Jayapura

Abstraksi - Sistem pengiriman data menjadi hal yang sangat penting akan tetapi informasi yang dikirim tidak sesuai dengan informasi yang diterima. Kesalahan dapat terjadi pada saat data tersebut ditransmisikan, hal ini disebabkan adanya gangguan jaringan yaitu seperti noise, kabel saluran sehingga data menjadi *error*. Sehingga membuat isi dari *file text* yang diterima berbeda/salah dengan isi *file* yang dikirim, yang diakibatkan oleh perubahan pada *bit-bit* data dari *file* tersebut. Adapun metode atau algoritma yang digunakan untuk aplikasi *error detection dan error correction* yaitu *parity* genap, *parity* ganjil, *haming code* dan menggunakan operasi gerbang logika *EX-OR (Exclusive-OR)*. Proses uji sistem yaitu menginputkan *file* yang diterima kemudian akan di konversikan ke bilangan *binary*, selanjutnya dideteksi apabila terjadi *error*, dan juga mengoreksi terletak diposisi berapa *bit* tersebut terdapat *error.* Metode *ood parity* (parity ganjil) dan *even parity* (parity genap), keduanya mampu mendeteksi lebih dari 1 buah *bit* yang *error* (2 bit), sedangkan algoritma *haming code* hanya bisa mendeteksi 1 buah *bit error* dan mengoreksi di posisi mana *bit* tersebut terdapat *error*.

Kata kunci : *bit, binary, error, error detection, error correction, EX-OR (Exclusive-OR), file, haming code, noise* dan *parity*.

1. PENDAHULUAN

Komunikasi menjadi sangat penting dan paling dibutuhkan bagi semua orang dalam kehidupan sehari-hari. Dengan adanya telekomunikasi semakin memudahkan seseorang untuk melakukan komunikasi diberbagai aktifitas/kegiatan pekerjaan sekalipun dengan jarak yang jauh pada media yang berbeda-beda dan bersifat efisien. Internet mampu memberikan fasilitas informasi dan komunikasi seperti *Google, Yahoo, facebook, twitter* dan masih banyak lagi. Komunikasi memberikan cara untuk menyebarkan informasi, data, ataupun berita dengan media transmisi yang dikenal dengan komunikasi data. Kepentingan komunikasi data terletak pada kalancaran proses penyampaian dan penyebaran data tersebut. Komunikasi data teridiri dari 2 objek yaitu pengirim dan penerima.

Komputer menjadi cara untuk mengirimkan data menggunakan sistem *transmitter* elektronik dari satu komputer ke terminal tertentu. Untuk menukar informasi, data ataupun berita, diperlukan sebuah akses agar mentransmisikan data ke sinyal yang akan ditukar ke dalam pesan. Pada media ini informasi atau pesan diubah kedalam bentuk kode biner yang melewati saluran tranmisi. Proses pengiriman pesan dapat terjadi gangguan yang menyebabkan berubahnya kode sehingga pesan yang diterima tidak sama dengan pesan yang dikirim. Perubahan seperti itu yang disebut dengan *error*. Untuk mengetahui terjadinya *bit error* ada beberapa metode yang bisa mendeteksi yaitu *parity*

genap dan *parity* ganjil, sedangkan algoritma *haming code* mampu mendeteksi sekaligus mengoreksi.

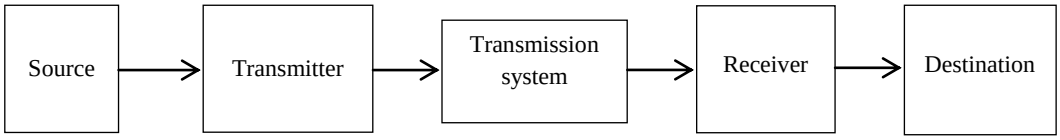
Tujuan penelitian ini untuk membangun sebuah aplikasi yang dapat mendeteksi dan mengoreksi jika terdapat kerusakan/*error* pada *file text* yang diakibatkan oleh adanya gangguan jaringan, *noise*, kabel dan lain-lain selama transmisi data berlangsung, sehingga isi dari *file* yang diterima bisa berbeda/salah dengan isi *file* yang dikirim, yang diakibatkan oleh perubahan pada *bit-bit* data saat proses transmisi terjadi.

2. TINJAUAN PUSTAKA

2.1 Komunikasi Data

Komunikasi data didefinisikan sebagai suatu cara untuk menyampaikan atau menyebarkan data, informasi atau berita menggunakan media transmisi data. Komunikasi cara yang dilakukan untuk mengirimkan data menggunakan sistem transmisi elektronik dari satu komputer ke komputer lain atau dari satu komputer ke terminal tertentu. Proses pengiriman data terjadi antara pihak pengirim (*transmitter*) dengan pihak penerima (*receiver*). *(Sumber : Ariyus dan Andi, 2008)*

Kerangka model komunikasi data digital dasar yang sederhana seperti gambar dibawah ini:



Gambar 2.1 : Kerangka komunikasi data *(Sumber: Ahmad Alfi Albar Lubis, 2011; 1)*

Keterangan :

a. *Source* (sumber) yaitu alat untuk membangkitkan data sehingga dapat ditransmisikan, contohnya komputer.
b. *Transmitter* melakukan konversi data ke bentuk signal analog untuk ditransmisikan, contohnya *modem* yang mentransformasikan *digital bit stream* yang diterima dari komputer menjadi *signal analog* melalui jaringan telepon.
c. *Source* dan *transmitter* merupakan bagian dari *transmission system*.
d. *Receiver* berfungsi untuk mengkonversi kembali signal analog yang diterima dari sistem transmisi menjadi data *digital*.
e. *Destination* menangkap data yang dihasilkan oleh *receiver* dan informasi dapat dibaca juga di *destination*.

2.2 Error Detection

Proses pelacakan kesalahan (*error*) selama transmisi data berlangsung, yaitu perubahan satu atau beberapa *bit* dari nilai “1” ke “0” atau sebaliknya yang terjadi saat pengiriman dan saat penerimaan oleh *receiver*. Kegiatan untuk memastikan bahwa data yang diterima sama dengan data yang dikirim. *Error* yang terjadi dapat disebabkan oleh kesalahan dalam transmisi (*hardware*), *network interface*, *noise* dan koneksi. *(sumber: Achmad Imam Kistijantoro, 2007;12)*

2.3 Hamming Code

Algoritma *Hamming Code* ditemukan oleh Richard W. Hamming pada tahun 1940-an. Algoritma *Hamming Code* merupakan salah satu algoritma pendeteksi *error* (*error detection*) yang mampu untuk mendeteksi beberapa error, namun hanya mampu mengoreksi satu *error* (single error correction). Algoritma pendeteksi *error* ini sangat cocok digunakan pada situasi dimana terdapat beberapa *error* yang teracak (*randomly occurring errors*). Algoritma *Hamming Code* menyisipkan beberapa buah *check bit* ke dalam data penerima. Jumlah *check bit* yang disisipkan tergantung pada panjang data. Algoritma *Hamming Code* merupakan algoritma *detection error* yang paling sederhana. Algoritma ini menggunakan operasi logika XOR (Exclusive – OR) dalam proses pendeteksian *error* (*error detection*) maupun proses pengkoreksian *error* (*error correction*),

sedangkan *input* dan *output* data dari algoritma *Hamming Code* berupa bilangan biner. Untuk data 2^n bit, jumlah *check bit* yang disisipkan ada sebanyak c = (n+1) *bit,* sehingga diperoleh table kenaikan data *bit* dan *check bit* seperti berikut:

Tabel 2.1 Kenaikan data bit dan *check bit*

Data Bit	Check Bit
2	2
4	3
8	4
16	5
32	6
64	7
128	8
256	9

(sumber: Ahmad Alfi Albar Lubis, 2011;2)

Check bit kemudian disisipkan pada data pada posisi yang dihitung menggunakan rumus perhitungan posisi check bit. Rumus perhitungan posisi *Check Bit* $C_i = 2^{i-1}$ sehingga dengan rumus posisi tersebut, didapat posisi *check bit* yang akan diletakkan pada data diperlihatkan pada table berikut :

Tabel 2.2 : Posisi *check bit*

Check Bit	Posisi
C_1	1
C_2	2
C_3	4
C_4	8
C_5	16
C_6	32
C_7	64
C_8	128
C_9	256

(sumber: Ahmad Alfi Albar Lubis, 2011;2)

Check bit ini yang digunakan untuk melakukan proses pendeteksi *error* (*error detection*) dan pengoreksi *error* (*error correction*). Algoritma proses pendeteksi *error* (*error detection*) dan pengoreksi *error* (*error correction*) dari algoritma *Hamming Code* adalah sebagai berikut:

1. Hitung panjang data *input* dari algoritma *Hamming Code* yang merupakan hasil penjumlahan dari panjang *input* data dan panjang *check bit*. Panjang data *output* dari algoritma *Hamming Code* sama dengan panjang data *input* dari algoritma *Hamming Code*.
2. Tandai semua posisi *bit* yang merupakan posisi dari *check bit*. Posisi selain posisi *check bit* merupakan posisi dari *data bit*.
3. Tentukan rumus perhitungan dari masing-masing *check bit*.
   Untuk n = 1 hingga jumlah dari *check bit*, lakukan hal berikut:
   a. Catat semua posisi dimana bit n dari *member position* bernilai 1, kecuali posisi bit itu sendiri. *Member position* merupakan bentuk biner dari posisi *bit*.
   b. Rumus dari *check bit* n sama dengan operasi *XOR* dari posisi – posisi yang dicatat.
4. Hitung nilai dari *check bit* untuk data *input* dan data *output*.

5. Jika nilai *check bit input* tidak sama dengan nilai *check bit output* berarti terdapat kesalahan (*error*).
6. Lakukan operasi *XOR* terhadap *check bit input* dan *check bit output*.
7. Konversikan hasil operasi *XOR* ke dalam bentuk bilangan desimal.
8. Jika nilai dari hasil operasi *XOR* lebih besar daripada panjang data *input* atau nilai dari hasil operasi *XOR* sama dengan posisi dari *check bit*, maka terdapat lebih dari satu kesalahan (*error*).
9. Jika tidak, maka jumlah kesalahan (*error*) hanya satu dan hasil operasi merupakan posisi data yang terdapat kesalahan (*error*).

2.4 Parity Check

Teknik *parity check* adalah bentuk teknik yang paling tua dalam system jaringan computer. Teknik ini dikenal sangat sederhana dalam melakukan deteksi dengan menambahkan *parity bit* pada *frame* yang dikirim. *Parity bit* digunakan dengan transmisi *asynchronous* sederhana. *Error* dideteksi dengan menambahkan sebuah *bit extra* yang disebut *bit parity,* di setiap ujung *frame. Bit* tambahan ini menjamin bahwa jumlah bit 1 yg ganjil dan yang genap dikirim di setiap transmisi.

Berdasarkan jumlah *bit-bit* "1" pada urutan *bit* yang disertainya, *bit parity* dibagi menjadi 2 jenis : *(sumber: Iwan Binanto, 2008;1)*

a. *Odd Parity* (Parity Ganjil) yaitu jika jumlah bit "1" dan *bit parity*nya adalah ganjil
b. *Even parity* (Parity Genap) yaitu jika jumlah bit "1" dan *bit parity*nya adalah genap

Tabel 2.3 : 7 *bit* data *even* dan *odd parity bit* :

7 bit data	Byte dengan bit parity	
	Even	ood
0000000	**0**0000000	**1**0000000
1010001	**1**1010001	**0**1010001
1101001	**0**1101001	**1**1101001
1111111	**1**1111111	**0**1111111

(sumber: Iwan Binanto, 2008;3)

Tabel 2.4 : tabel kebenaran *Odd Parity Bit* yang dibangkitkan dari urutan data 3 *bit* biner (ABC)

INPUT			OUTPUT
A	B	C	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(sumber: Iwan Binanto, 2008;3)

Tabel 2.5 : tabel kebenaran Even Parity Bit yang dibangkitkan dari urutan data 3 bit biner (ABC)

INPUT			OUTPUT
A	B	C	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(sumber: Iwan Binanto, 2008;3)

3. HASIL DAN PEMBAHASAN

Pada saat data berada dalam transmisi kemungkinan sistem terdapat data yang *error*. Masalah yang harus dihadapi dalam sistem komunikasi adalah terajadinya *error* yang menyebabkan sistem tersebut tidak sesuai dengan yang diinginkan. Hal ini disebabkan karena adanya gangguan pada *signal (noise)* pada saat data ditransmisikan melalui sistem tranmisi sehingga pesan atau data yang diterima berubah ketika sampai di *receiver*. Selama pengiriman data baik berupa *signal digital* maupun *signal* analog data yang dikirim berupa *bit* dapat mengalami perubahan dari nilai "1" ke "0" atau sebaliknya. Misalkan data yang dikirim kata "HALO" dimana "HALO" = "0100 **1**000 0100 0001 0100 1100 0100 1111" sedangkan data yang diterima menjadi "0100 **0**000 0100 0001 0100 1100 0100 1111 ", sehingga diketahui letak dan posisi dimana bit mengalami perubahan *error*.

a. *Parity* Genap (*Even Parity*)

Kasus pertama :
Contoh *file text* yang akan dikirim yaitu "HALO" dengan binary sebagai berikut :

0100 1000 0100 0001 0100 1100 0100 1111
H A L O

Tabel 3.1 Gerbang Logika *Exlusive-OR (XOR)*

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

si-A menambahkan *bit* "0" karena *bit* 1 berjumlah genap dan dikirim :

0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 1 1 **0**
si-A menghitung jumlah nilai *bit parity* :
= **0**

si-B menerima : 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 1 1 **0**

si-B menghitung keseluruhan *parity* :

0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 1 1 1 **0**
= **0** (benar)

si-B melaporkan bahwa transmisi data berhasil dengan hasil *parity* yang benar/sama.

Kasus kedua :

Contoh *file text* yang akan dikirim yaitu “AYAH” dengan binary sebagai berikut :

0100 0001	0101 1001	0100 0001	0100 1000
A	Y	A	H

si-A menambahkan *bit* “0” karena *bit* 1 nya berjumlah genap dan dikirim :
0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 **0**
si-A menghitung jumlah nilai *bit parity* :
= **0**

si-B menerima :
0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 ***0*** 0 0 0 **0**

TRANSMISSION ERROR terdapat perubahan *bit*

si-B menghitung keseluruhan parity :
0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 ***0*** 0 0 0 **0**
= **1** (**salah**)

si-B melaporkan terjadi kesalahan transmisi, karena hasil nilai parity tidak sama.

b. Parity Ganjil (Odd Parity)

Kasus pertama :

Contoh *file text* yang akan dikirim yaitu “BAJU” dengan binary sebagai berikut :

0100 0010	0100 0001	0100 1001	0101 0101
B	A	J	U

si-A menambahkan *bit* “0” karena *bit* 1 nya berjumlah ganjil dan dikirim :

0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 **0**

si-A menghitung jumlah nilai *bit parity* :

= **0**

si-B menerima :

0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 **0**

si-B menghitung keseluruhan *parity* :

0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 **0**

= **0** (benar)

si-B melaporkan bahwa transmisi berhasil dengan parity yang benar/sama.

Kasus kedua :

Contoh *file text* yang akan dikirim yaitu “APEL” dengan binary sebagai berikut :

0100 0001 (A) 0101 0000 (P) 0100 0101 (E) 0100 1100 (L)

si-A menambahkan *bit* “1” agar berjumlah ganjil, karena *bit* 1 nya berjumlah ganjil dan dikirim :

0 1 0 0 0 0 0 1 0 1 0 1 0 00 0 0 1 0 0 0 1 0 1 0 1 0 0 1 1 0 0 **1**

si-A menghitung jumlah nilai *bit parity* :

= **1**

si-B menerima :

0 1 0 0 0 0 0 1 0 1 0 1 0 00 0 0 1 0 0 0 ***0*** 0 1 0 1 0 0 1 1 0 0 **1**

**TRANSMISSION ERROR* terdapat perubahan *bit*

si-B menghitung keseluruhan parity :

0 1 0 0 0 0 0 1 0 1 0 1 0 00 0 0 1 0 0 0 ***0*** 0 1 0 1 0 0 1 1 0 0 **1**

= **0** (**salah**)

si-B melaporkan terjadi kesalahan transmisi, karena hasil nilai parity tidak sama.

c. Haming Code

Pengiriman pesan dengan menggunakan kata “HALO”
PENYISIPAN *CHECK BIT*
Data masing-masing karakter terlebih dahulu diubah ke dalam kode ASCII kemudian di *XOR*kan.
HALO = 4 Karakter
1 Karakter= 1 *Byte* = 8 *bit*

0100 1000 (H) 0100 0001 (A) 0100 1100 (L) 0100 1111 (O)

Panjang 1 kata 4 karakter (HALO) yaitu 32 bit = **0100 1000 0100 0001 0100 1100 0100 1111**
Untuk data 2^n *bit*
32 *bit* = 2^5 *bit*
Maka jumlah *Check Bit* yang disisipkan sebanyak c = (n + 1)
C = 5+1 = 6 *bit*
Sehingga panjang *bit* yang akan dikirim, 32+6 bit = 38 *bit*

Tabel 3.2 Panjang *bit* data

Pangkat dua	Panjang data
2^0	0
2^1	1
2^2	4
2^3	8
2^4	16
2^5	32

Tabel 3.3 : Gerbang Logika *Exlusive-OR (XOR)*

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Hitung nilai total 38 *check bit* :

		0		1	0	0		1	0	0	0	0	1	0		0	0	0	0	1	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

1	0	0	1	1	0	0	0	1		0	0	1	1	1	1
23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38

C1 = 0 1 0 1 0 0 0 0 0 1 1 0 1 0 1 0 1 1 = **0**
C2 = 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1 = **1**
C4 = 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 = **1**
C8 = 1 0 0 0 0 1 0 0 1 1 0 0 0 1 = **1**
C16 = 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 = **1**
C32 = 0 0 1 1 1 1 = **0**

0	**1**	0	**1**	1	0	0	**1**	1	0	0	0	0	1	0	**1**	0	0	0	0	1	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

1	0	0	1	1	0	0	0	1	**0**	0	0	1	1	1	1
23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38

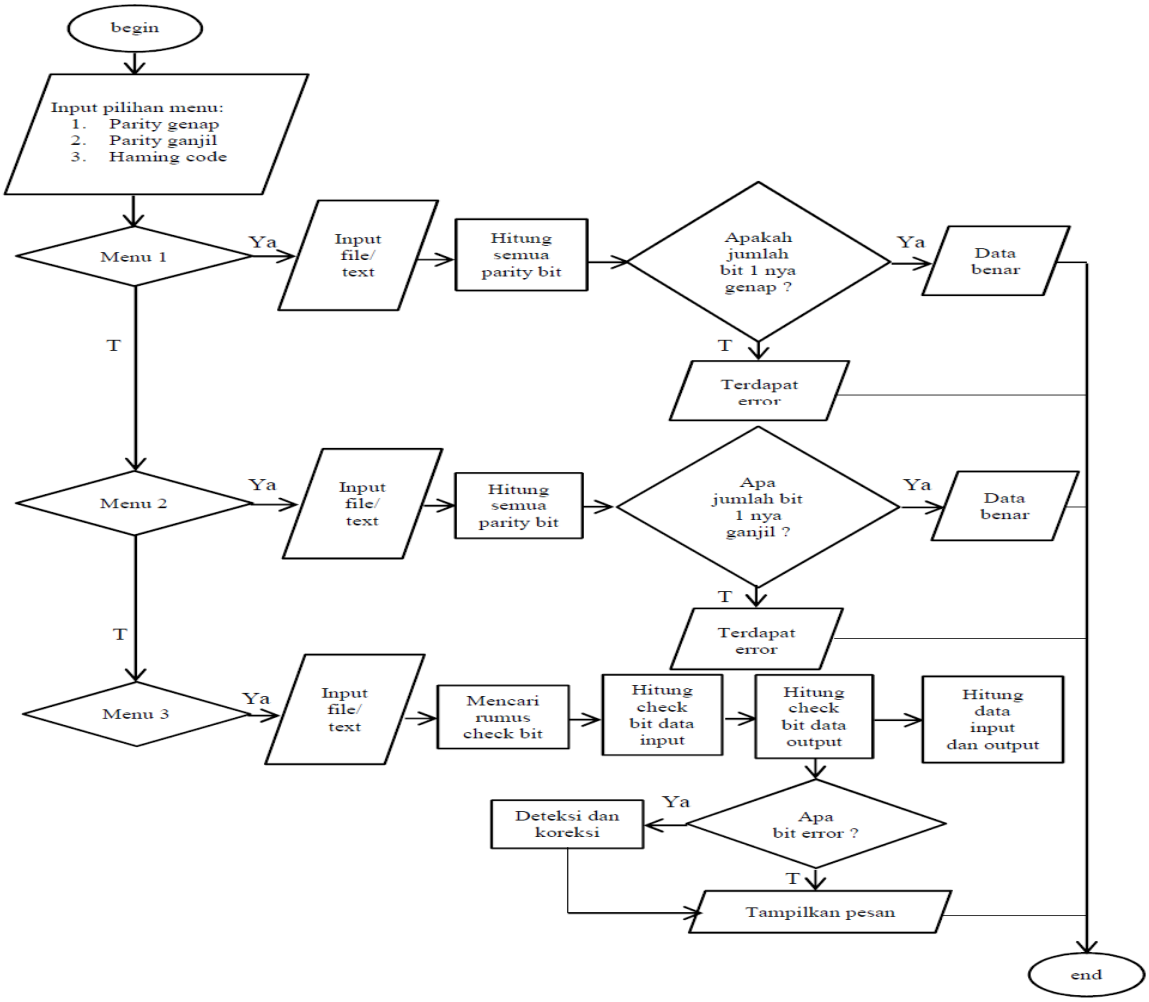
Hasilnya : 0101 1001 1000 0101 0000 1010 0110 0010 0011 11
PENDETEKSI *ERROR*
Panjang bit yang diterima = 38
Original bit = 32
Check bit = 6
Hasilnya:
Check Bit yang didapat = 0 1 1 1 1 0
Check Bit yang di ekstrak = 0 1 1 1 1 0
Sama berarti tidak terdapat kesalahan pada pengiriman_data.

Contoh terdapat kesalahan:

Check Bit yang didapat = 0 1 0 1 1 0
Check Bit yang di ekstrak = 0 1 1 1 1 0
Tidak sama berarti terdapat kesalahan pada pengiriman_data.
Check Bit yang didapat = 0 1 0 1 1 0
Check Bitt yang di ekstrak = 0 1 1 1 1 0

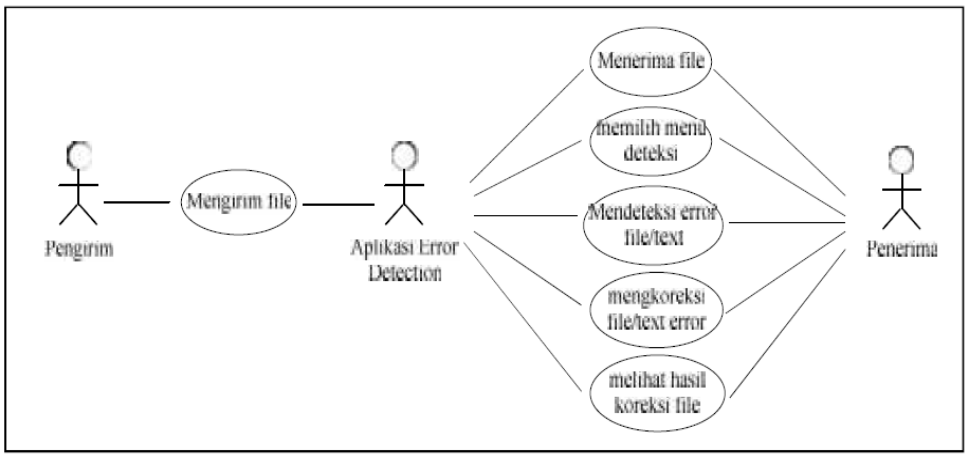
Hasil XOR = 0 0 1 0 0 0
001000 = 8
Hanya terdapat satu kesalahan karena 8 < dari panjang data =32 *bit*, dan samadengan posisi *check bit*. Kesalahan hanya pada satu posisi yaitu *bit* ke 8 dari inputan
Contoh *bit* yang salah : 0101 100**0** 1000 0101 0000 1010 0110 0010 0011 11
Bit ke8 = 0 Dinegasikan = 1
Barisan *bit* yang benar = 0101 100**1** 1000 0101 0000 1010 0110 0010 0011 11

d. Flowchart



Gambar 3.1 Flowchart System

e. Use Case Diagram



Gambar 3.2 Use Case Diagram

f. Menu "Odd Parity Check"

Gambar 3.3 Tampilan Menu "Odd Parity Check"

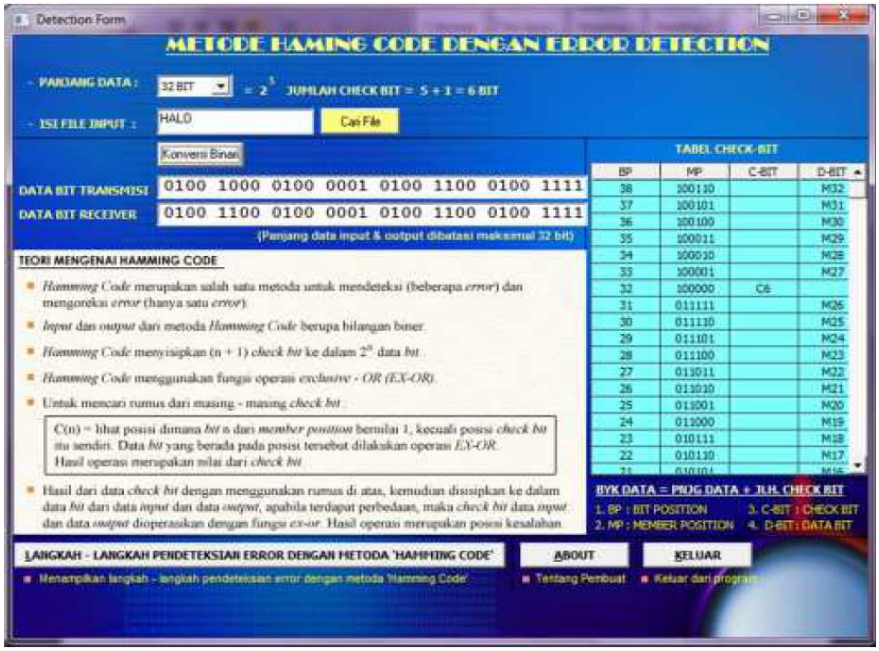
Dapat mencari *file text*, dapat menampilkan isi *file text*, dapat mengkonversi *binary* serta menambahkan *bit*, mampu menghitung data *parity* ganjil *bit* transmisi, menghitung data *parity* ganjil bit transmisi *receiver* dan menampilkan informasi berupa pesan tidak *error*.

g. Menu "Even Parity Check"

Gambar 3.4 Tampilan Menu "Even Parity Check"

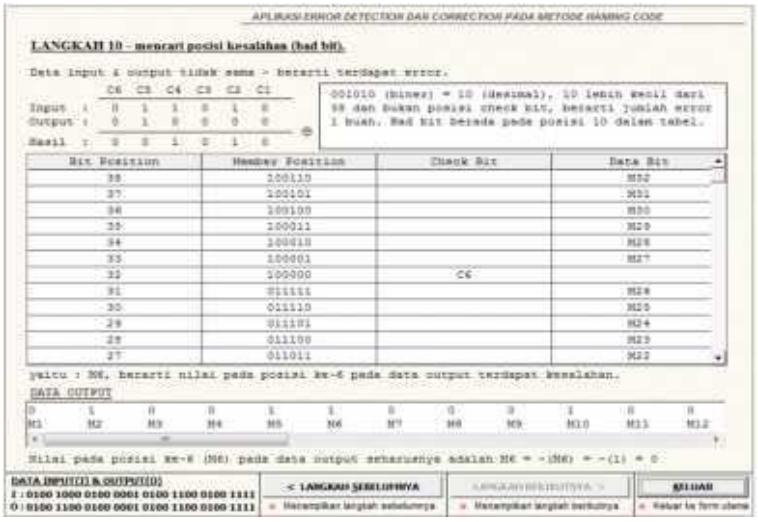
Dapat mencari *file text*, dapat menampilkan isi *file text*, dapat mengkonversi *binary* serta menambahkan *bit*, mampu menghitung data *parity* genap *bit* transmisi, menghitung data *parity* genap bit transmisi *receiver* dan menampilkan informasi berupa pesan *error*.

h. Menu "Hamming Code"



Gambar 3.5 Tampilan Menu "Metode Hamming"

Dapat memilih panjang data dan melihat rumus *check bit* yang kemudian ditampilkan pada tabel *check bit*, mencari *file* dan menampilkan isi dari *file text*, dapat dikonversikan ke *binary* dan menampilkan data transmisi serta *receiver* dan selanjutnya dapat melakukan proses deteksi dan koreksi *file text*.



Gambar 3.6 Tampilan Hasil Deteksi dan Koreksi Error

Dapat menampilkan informasi hasil deteksi data *input* dan *output*, dan informasi kesalahan posisi *bit* yang *error*.

4. PENUTUP

a. Kesimpulan

Dengan beberapa pilihan metode/algoritma dapat diketahui bagaimana cara mendeteksi dan mengoreksi kerusakan/*error* pada file text yang diakibatkan oleh adanya gangguan jaringan berupa noise, kabel dan lain-lain selama transmisi data berlangsung. Sehingga isi dari file yang diterima berbeda/salah dengan isi file yang dikirim, yang diakibatkan oleh perubahan pada bit-bit data dari file tersebut.

Metode *odd parity* (parity ganjil) dan *even parity* (parity genap), keduanya mampu mendeteksi lebih dari 1 buah *bit* yang *error* (2 bit), sedangkan algoritma *haming code* hanya bisa mendeteksi 1 buah *bit error* dan mengoreksi di posisi mana *bit* tersebut terdapat *error*.

b. Saran

Dalam hal ini diberikan beberapa saran pada tugas akhir yang mungkin berguna untuk pengembangan lebih lanjut pada perancangan aplikasi *error detection* dan *error correction* antara lain:

1. Deteksi *bit check in error* juga dapat dikembangkan selain *file text* misalkan data *image* atau yang lainnya.
2. Untuk kedepannya aplikasi bisa di uji coba dengan menggunakan 2 perangkat komputer yaitu sebagai pengirim dan penerima serta alat percobaan dalam tes pengiriman data/*file text* misalkan *Bluetooth, wi-fi, e-mail* dan lain-lain.
3. Menggunakan lebih banyak karakter pada isi *file text* misalkan terdiri dari 1 kalimat ataupun 1 paragraf.

5. DAFTAR PUSTAKA

Deri, D.P., 2013, Perancangan Aplikasi Deteksi Bit Check In Error Pada Transmisi Data Text Dengan Single Error Correction Menggunakan Algoritma Hamming Code, *Pelita Informatika Budi Darma*, 3, IV, http://pelita-informatika.com/berkas/jurnal/4321.pdf, 18 September 2014.

Dwiwulandari, B., 2008, Aplikasi Kode Haming Sebagai Error Detecting Code Dalam Pengiriman Pesan, *Skripsi Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia,* Depok.

Fitri, 2008, Analisis Algoritma Dan Waktu Dekoding Kode Bch Dalam Pengoreksian Galat Pada Transmisi Pesan Teks**,** *Skripsi Departemen Ilmu Komputer, Fakultas Matematika Dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor,* Bogor.

Lubis Ahmad Alfi Albar, Sihombing Poltak, Sani Arman, 2011, Perancangan Error Detection System And Error Correction System Menggunakan Metode Hamming Code Pada Pengiriman Data Text, *Jurnal, Program Studi S1 Ilmu Komputer, Universitas Sumatera Utara,* Medan.

Maharani Tamara, Pratiarso Aries, Arifin, 2011, Simulasi Pengiriman Dan Penerimaan Informasi Menggunakan Kode Bch, *jurnal Politeknik Elektronika Negeri Surabaya Institut Teknologi Sepuluh November*, Surabaya.

Pratiwi Swelandiah Endah, Kurniawati Anna, 2012, Algoritma Perhitungan Langsung Pada Cyclic Redundancy Code 32, *Jurnal*, Fakultas Ilmu Komputer Dan Teknologi Informasi, Universitas Gunadarma, Jakarta.

Puspita, Dantik., 2013, Menghitung bit paritas dan mendeteksi kesalahan haming code, *http://www.youtube.com/watch?v=YOfbPlWQVbU*, 20 Oktober 2014.

Sutarto Mohammad, Suwadi, Suryani Titiek, 2014, Implementasi *Encoder* Dan *Decoder* Bch menggunakan DSK TMS320C6416T, *Jurnal Jurusan Teknik Elektro, Institut Teknologi Sepuluh November,* Surabaya.

Setiawan, I., 2013, Komunikasi Data, Jaringan & Internet, *http://stwn.blog.unsoed.ac.id/files/2013/04/komdat-2013-1.pdf*, 28 Desember 2014.

Tanenbaum, A. S., Watherall, D. J., 2011, *Computer Network*, edisi kelima, Pearson Education, Inc., Boston.