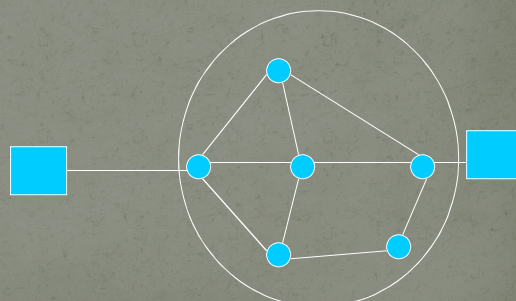


Jaringan Komputer

Materi 4
Lapis Datalink

Terminologi Fisik Jaringan

Node ●
Link —
Terminal ■
Jaringan ○



Link

- Jalur yang menghubungkan antar 2 elemen jaringan (node-node atau terminal-node)
- Kumpulan link (+ node-node) = jaringan
- Fungsi link sangat vital, maka OSI menetapkan protokol lapis 2 (datalink)
- Datalink = mengatur agar komunikasi di link tersebut berjalan *benar* dan *lancar*
- Tidak ada keharusan jenis link dalam jaringan sama = boleh memilih teknologi link (fisik maupun protokol) untuk setiap link
- Terdapat 2 macam link : link fisik dan link logik (contoh: virtual path yang terdiri atas virtual channel)

Tugas Datalink

- Pembukaan hubungan dan penutupan hubungan
- Melakukan kendali atas kesalahan yang mungkin terjadi : tool → pariti, crc, dll
- Melakukan pengendalian banyaknya data yang dikirim → untuk menghindari kemacetan (kongesti) : tool → sliding windows dll
- Dan lainnya (optional : tambahan untuk protokol datalink tertentu)

Proses Hubungan Di Link

- Ada 2 jenis proses hubungan di link :
 - Memerlukan connection setup
 - Hubungan langsung
- Connection setup
 - Ada banyak path yang bisa dipilih
 - Untuk hubungan yang sangat handal
 - Tersedia berbagai pilihan kecepatan komunikasi
- Hubungan langsung
 - Tanpa pilihan jalur dan kecepatan komunikasi
 - Point-to-point connection

Metoda Deteksi Kesalahan

- Agar bisa melakukan kendali kesalahan, syarat mutlak yang harus ada adalah adanya mekanisme deteksi kesalahan
- Beberapa metoda yang umum digunakan:
 - Pariti → paling sederhana
 - CRC → lebih sulit, meminta kemampuan komputasi
 - Checksum → operasi word

Pariti

- Penambahan 1 bit sebagai bit deteksi kesalahan
- Terdapat 2 jenis pariti : genap dan ganjil
 - Pariti genap = jumlah bit 1 dalam kode adalah genap
 - Pariti genap = $d_1 \text{ xor } d_2 \text{ xor } \dots \text{ xor } D_n$
 - Pariti ganjil = jumlah bit 1 dalam kode adalah ganjil
 - Pariti ganjil = $(d_1 \text{ xor } d_2 \text{ xor } \dots \text{ xor } D_n) \text{ xor } 1$
- Sistem sederhana dan mudah dibuat hardwarenya (di PC digunakan IC 74LS280)
- Tidak mampu mendeteksi kesalahan bit genap → peluang benar deteksi kesalahan hanya 25% (peluang salah mutlak 50% + peluang salah deteksi 25%)

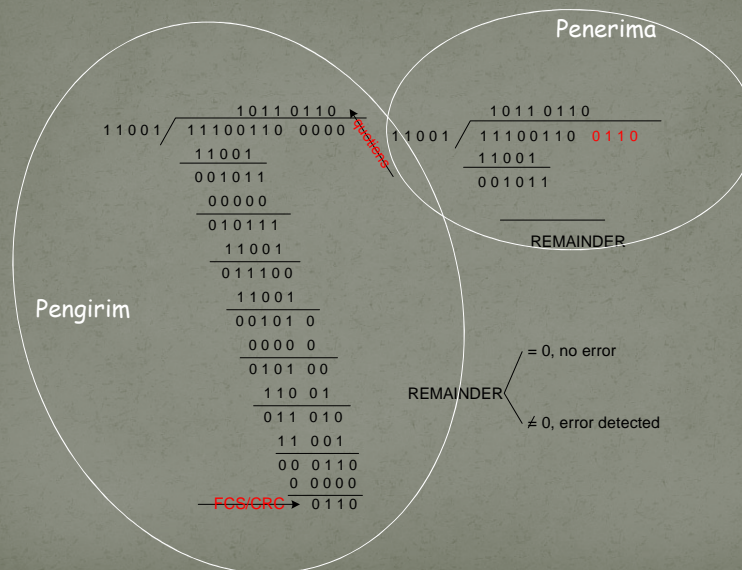
Cyclic Redundancy Check: Sisi Pengirim

- Merupakan hasil operasi pembagian biner dengan suatu pembagi tertentu (generator polinomial)
 - Pembagi : $D_n D_{n-1} \dots D_1$
 - Deretan bit : $b_1 b_2 b_3 \dots b_m$
 - Operasi :
 - $(b_1 b_2 b_3 \dots b_m)_{n-1} / D_n \dots D_1 \rightarrow \text{sisa } (R_{n-1} \dots R_1)$
 - Dikirim $b_1 b_2 b_3 \dots b_m R_{n-1} \dots R_1$

Cyclic Redundancy Check: Sisi Penerima

- Oleh penerima dilakukan operasi yang sama
 - $b_1 b_2 b_3 \dots b_m R_{n-1} \dots R_1 / D_n \dots D_1 \rightarrow \text{sisa } (r_{n-1} \dots r_1)$
 - Data benar jika $r_{n-1} \dots r_1 = 0$
 - Data salah jika $r_{n-1} \dots r_1 \neq 0$
- Pembagi standar internasional
 - CRC-16 $\rightarrow 11000000000000101$
 - CRC-ITU $\rightarrow 10001000000100001$
 - CRC-32 $\rightarrow 10000010010000001000111011011011$
- Jika diperlukan pembagi boleh tidak menggunakan standar ini asal memenuhi:
 - Diawali dan diakhiri dengan bit 1 (1xxxxxx1)
 - Jumlah minimum bit "1" : 3 bit
 - Agar bisa mendeteksi jumlah bit kesalahan ganjil : harus habis dibagi oleh $(11 = X + 1)$

Contoh Perhitungan CRC



Penggunaan : Pada Paket LAN (MAC)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Destination MAC Address
Source MAC Address
Protocol/Length
Data (46 – 1500 B)
CRC-32

Checksum

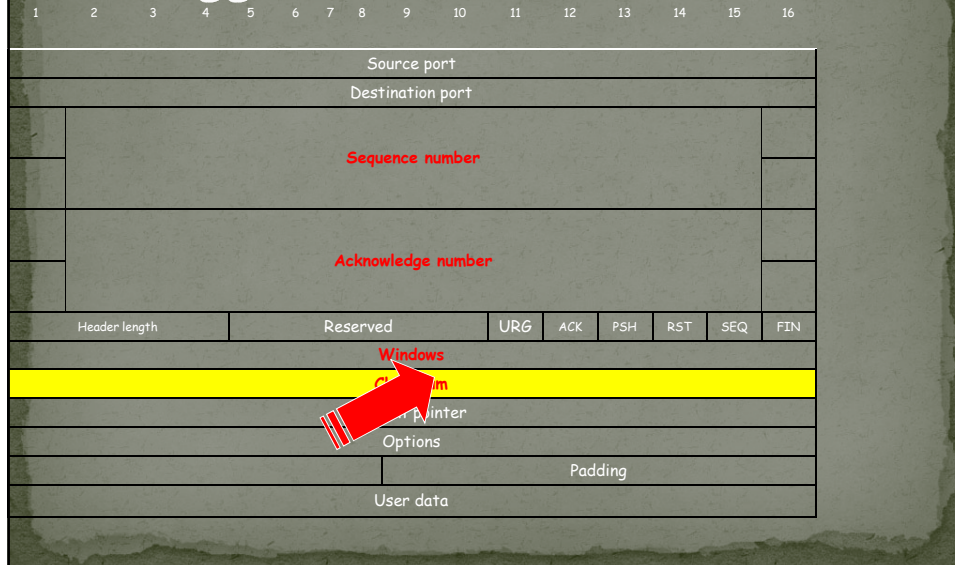
- CRC memerlukan perhitungan xor sebanyak jumlah bit data → memerlukan kemampuan komputasi yang cukup besar
- Diciptakan metoda checksum (untuk mengurangi perhitungan) pada beberapa jenis transmisi tidak perlu kecanggihan CRC atau sudah melakukan CRC di lapis lain
- Cara perhitungan checksum:
 - Data dibagi menjadi kelompok-kelompok 16 bit (word)
 - Word pertama di xor dengan word kedua
 - Hasil di xor dengan word ketiga, keempat, ...sampai word terakhir (jika bit-bit terakhir tidak cukup untuk menjadi word, ditambahkan padding bit '0' sampai membentuk word)
 - Hasil akhir (16 bit) = checksum

Contoh perhitungan

Pengguna Checksum: IP

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
								Priority (0-7)		low		high		high		← "1"
Version				Header length (dword)				Precedence		D		T		R		unused
Total length																
Identification																
D		M				Fragment offset										
Time to live (seconds)								Protocol								
Header checksum																
Source IP address																
Destination IP address																
Option (0 word atau lebih)																
Data ≤ 64 kB																

Pengguna Checksum: TCP



Backward Error Control

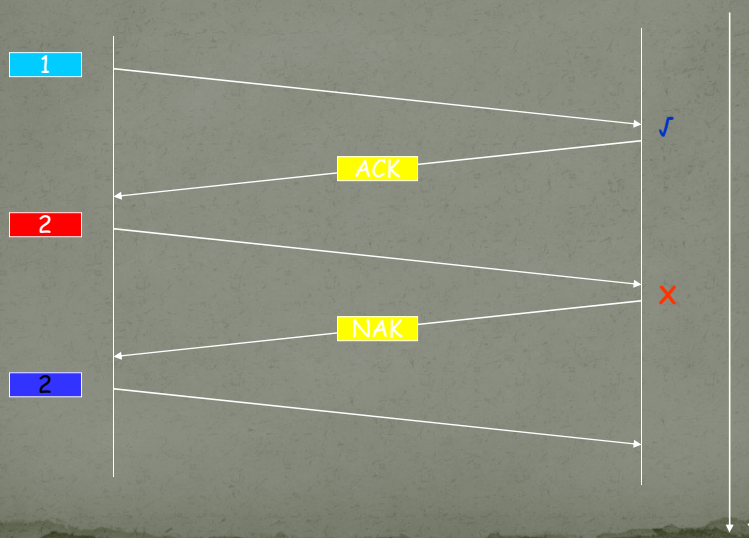
- Kemampuan deteksi kesalahan digunakan untuk melakukan perbaikan kesalahan (error control) dengan cara meminta pengiriman ulang jika paket yang diterima salah



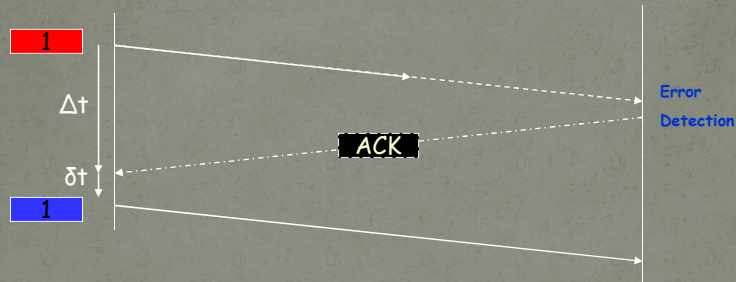
Backward Error Control: ARQ

- ARQ = Automatic ReQuest
- ARQ akan mengulang / tidak mengulang pengiriman data sesuai dengan feedback dari penerima
 - ACK = acknowledge → data diterima benar
 - NAK = not acknowledge → data diterima salah
- Feedback dari penerima

ARQ : Idle RQ



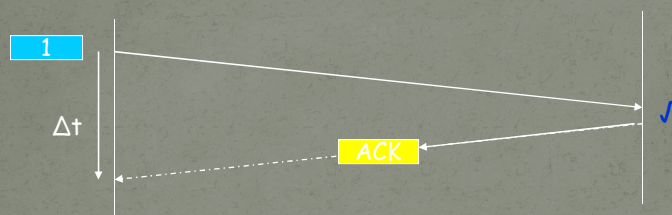
Kasus 1: jika paket tidak sampai



Pengirim menunggu feedback sampai $\Delta t + \delta t$, jika tidak ada respon maka pengirim harus mengirimkan kembali paket tersebut.

Waktu tersebut disebut dengan waktu *timeout*

Kasus 2: feedback tidak sampai



Diperlakukan sama dengan kondisi kasus 1 (time-out)

Kapankah pengirim mengirim ulang paket ???

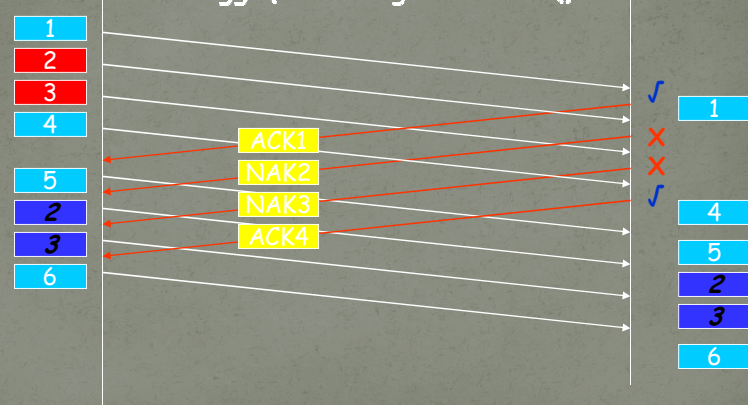
- Jika mendapat feedback NAK
- Jika timeout
- Jika mendapat feedback yang tidak dimengerti
- Kesimpulan : pengirim mengirim ulang paket
→ Jika **tidak** mendapat ACK

ARQ : Idle RQ

- "**DIE HARD/Persistent/Ngotot**" ARQ
- Paket akan diterima terjaga urutannya
- Efisiensi saluran paling rendah
- Cocok digunakan untuk saluran transmisi yang sangat jelek kualitasnya (banyak error)

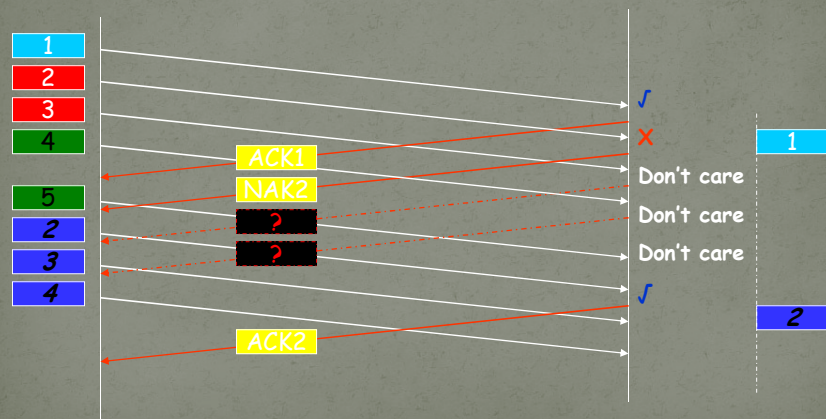
ARQ : Selective Repeat

- Hanya mengirim ulang untuk paket yang salah
- Paket diterima tidak berurutan
- Efisiensi saluran tinggi (dibandingkan idle RQ)

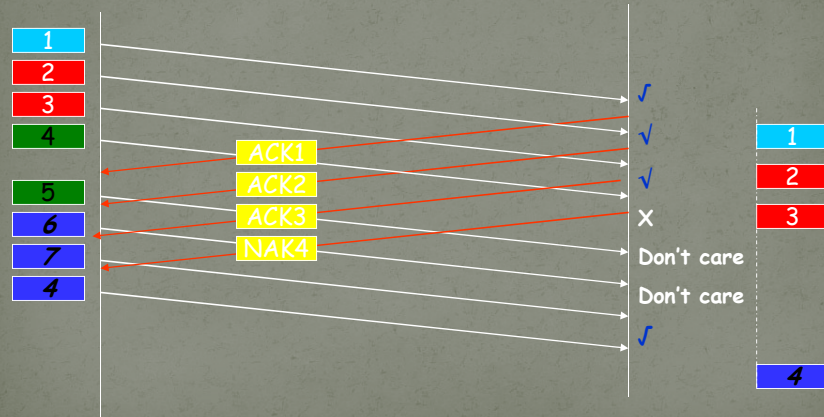


ARQ : Go Back N

- Mengirim ulang mulai dari paket yang salah
- Paket akan diterima terjaga urutannya
- Efisiensi saluran lebih rendah dari Selective Repeat



Kasus Lain Go Back N



Error Control 1 (Backward Error Control)

- Error control = error detection + ARQ
- Kelemahan : waktu yang diperlukan untuk mengirim dengan benar adalah minimal 2 x waktu propagasi

Forward Error Control

- Backward EC menyebabkan delay pengiriman paket yang cukup besar tergantung dari berapa kali paket tersebut harus dikirim
- Untuk sistem transmisi jarak jauh dimana delay propagasi sangat besar (kelas detik, menit atau jam) BEC tidak bisa menjadi pilihan
- Juga untuk aplikasi multimedia, dimana ketepatan waktu kedatangan lebih utama dibandingkan dengan 'kebenaran' data, BER menyebabkan delay yang lewat batas toleransi waktu
- Dipergunakan Forward Error Correction (FEC) untuk memecahkan masalah ini
- FEC berprinsip dasar: penerima mampu membetulkan sendiri kesalahan data yang sudah diterima, karena selain menerima data juga menerima bit-bit redundansi yang diperlukan

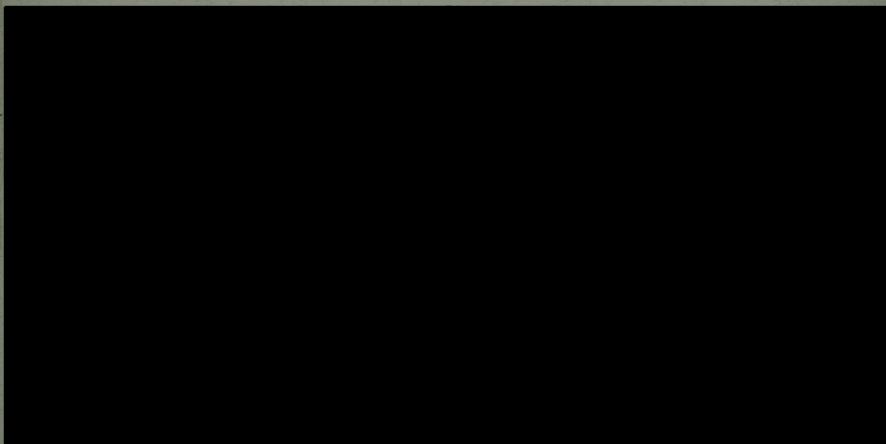
Jenis-Jenis FEC

- Metoda FEC yang umum dikenal :
 - Block Parity
 - Hamming Code
 - Turbo Code, RS Code, BCH Code
- Block Parity
 - Sederhana, menggunakan perhitungan pariti dasar
 - Menggunakan pariti baris dan kolom sebagai sarana koreksi kesalahan
 - Hanya mampu **mengkoreksi** kesalahan 1 bit, mampu **mendeteksi** kesalahan lebih dari 1 bit
 - Efisiensi tergantung dari ukuran baris dan kolom yang digunakan, semakin banyak baris dan kolom akan semakin banyak bit pariti

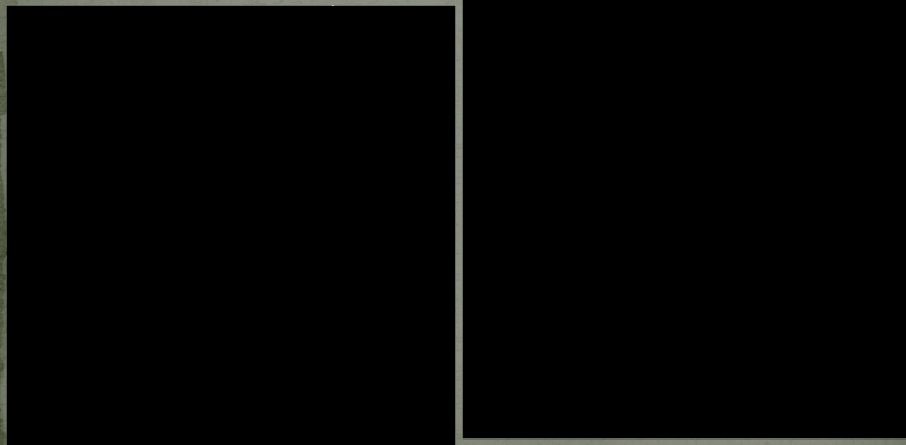
Contoh Block Parity 1



Contoh Block Parity 2



Contoh Block Parity 3



Hamming Code: Sisi Pengirim

- Menggunakan metoda matematik modulo 2
- Disisipkan bit-bit pariti di posisi bit 2^n : bit ke 1,2,4,8,16,32 dst
- Bit pariti dihitung dengan cara:
 - $P_1 = d_1 \text{ xor } d_2 \text{ xor } d_4 \text{ xor } d_5 \text{ xor } d_7 \text{ xor } d_9 \text{ dst}$
 - $P_2 = d_1 \text{ xor } d_3 \text{ xor } d_4 \text{ xor } d_6 \text{ xor } d_7 \text{ xor } d_{10} \text{ dst}$
 - $P_3 = d_2 \text{ xor } d_3 \text{ xor } d_4 \text{ xor } d_8 \text{ xor } d_9 \text{ xor } d_{10} \text{ dst}$
 - $P_4 = d_5 \text{ xor } d_6 \text{ xor } d_7 \text{ xor } d_8 \text{ xor } d_9 \text{ xor } d_{10} \text{ dst}$
 - $P_5 = d_{12} \text{ xor } d_{13} \text{ xor } d_{14} \text{ xor } d_{15} \text{ dst}$
- Banyaknya bit pariti yang dibutuhkan tergantung jumlah bit datanya
- Sehingga deretan bit → $P_1 P_2 d_1 P_3 d_2 d_3 d_4 P_4 d_5 d_6 d_7 d_8 d_9 \text{ dst}$ untuk ditransmisikan

0	0	0	1	P ₁	Machine Way
0	0	1	0	P ₂	
0	0	1	1	D ₁	
0	1	0	0	P ₃	
0	1	0	1	D ₂	P ₁ = d ₁ xor d ₂ xor d ₄ xor d ₅ xor d ₇
0	1	1	0	D ₃	dst
0	1	1	1	D ₄	P ₂ = d ₁ xor d ₃ xor d ₄ xor d ₆ xor d ₇
1	0	0	0	P ₄	dst
1	0	0	1	D ₅	P ₃ =
1	0	1	0	D ₆	P ₄ =
1	0	1	1	D ₇	
1	1	0	0	D ₈	
1	1	0	1	D ₉	
1	1	1	0	D ₁₀	

Human Way

- P₁ P₂ 1 P₃ 1 0 1 P₄ 1 0 1 0 1 1 1

0	0	1	1	3
0	1	0	1	5
0	1	1	1	7
1	0	0	1	9
1	0	1	1	11
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

1	1	1	1
P ₄	P ₃	P ₂	P ₁

- 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1

Hamming Code: Sisi Penerima

- Setelah diterima dilakukan perhitungan
 - $H1 = P1 \text{ xor } d1 \text{ xor } d2 \text{ xor } d4 \text{ xor } d5 \text{ xor } d7 \text{ xor } d9 \text{ dst}$
 - $H2 = P2 \text{ xor } d1 \text{ xor } d3 \text{ xor } d4 \text{ xor } d6 \text{ xor } d7 \text{ xor } d10 \text{ dst}$
 - $H3 = P3 \text{ xor } d2 \text{ xor } d3 \text{ xor } d4 \text{ xor } d8 \text{ xor } d9 \text{ xor } d10 \text{ dst}$
 - $H4 = P4 \text{ xor } d5 \text{ xor } d6 \text{ xor } d7 \text{ xor } d8 \text{ xor } d9 \text{ xor } d10 \text{ dst}$
 - $H5 = P5 \text{ xor } d12 \text{ xor } d13 \text{ xor } d14 \text{ xor } d15 \text{ dst}$
- Jika disusun menjadi H5 H4 H3 H2 H1 dan terbaca :
 - 00000 = 0 → tidak ada kesalahan
 - 00101 = 5 → bit 5 (d2) salah
 - 01001 = 9 → bit 9 (d5) salah

Human Way

- 111110111010111
- Karena $H4H3H2H1 = 0000 \rightarrow$ diterima benar

0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	1	11
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
0	0	0	0	
H4	H3	H2	H1	

Human Way

- 111111111010111
- Karena $H_4H_3H_2H_1 = 0110$ (6) → yang salah bit ke 6
- 111111111010111
→ 1111011101011

0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	1	11
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
0	1	1	0	
H ₄	H ₃	H ₂	H ₁	

Human Way

- 111100111010111
- Karena $H_4H_3H_2H_1 = 0101$ (5) → yang salah bit ke 5
- 111100111010111
→ 1111011101011

0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	1	11
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
0	1	0	1	
H ₄	H ₃	H ₂	H ₁	

Human Way

111100011010111

- Karena $H_4H_3H_2H_1 = 0010 \rightarrow$ bit 2 salah

- Katanya yang benar adalah

101100011010111

- Hanya mampu mengkoreksi 1 bit error

0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
1	0	0	0	8
1	0	0	1	9
1	0	1	1	11
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
0	0	1	0	
H_4	H_3	H_2	H_1	

- Objektif: dua code word yg dibolehkan akan mempunyai jarak minimum (minimum distance) d
 - Dapat mendeteksi sampai dg $d - 1$ error
 - Dapat mengkoreksi sampai dg $d/2$ error

Metoda FEC Lain

- Semua metoda FEC pada dasarnya menggunakan metoda matematik modulo 2
- Metoda ini terus dikembangkan dengan tujuan:
 - Mendapatkan kemampuan koreksi bit yang semakin banyak
 - Dengan mengurangi jumlah bit pariti yang dibutuhkan
 - Mampu melanjutkan komunikasi walaupun sempat terputus.
- Metoda yang umum digunakan:
 - BCH Code
 - Reed Solomon Code
 - Convolutional Code
 - Trellis Code
 - Turbo Code

- 11001011100
- A. hamming nya
- B. bit ke 6 error
- C. bit 7 & 9 error