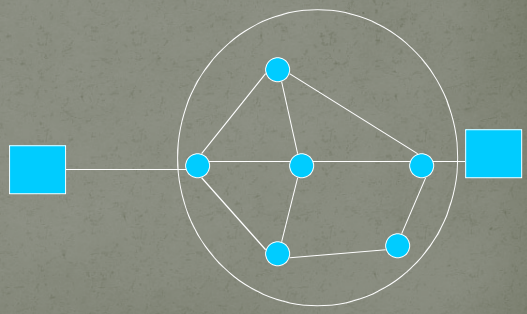
Jaringan Komputer

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 = d1 xor d2 xor ..... Dn
  - Pariti ganjil = jumlah bit 1 dalam kode adalah ganjil
  - Pariti ganjil = (d1 xor d2 xor ..... Dn) xor 1
- Sistem sederhana dan mudah dibuat hardwarenya (di PC digunakan IC 74LS280)

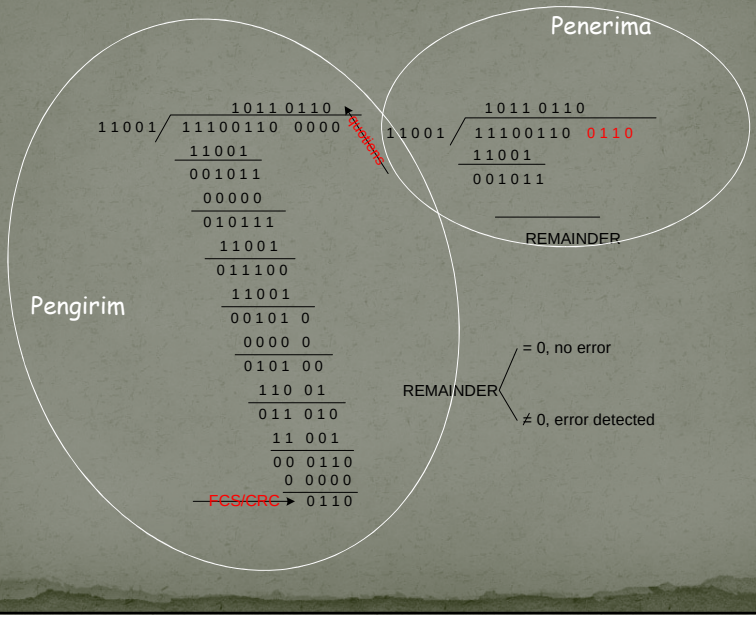
Cyclic Redudancy Check: Sisi Penggirim

- Merupakan hasil operasi pembagian biner dengan suatu pembagi tertentu (generator polinomial)
  - Pembagi : Dn Dn-1 ...D1
  - Deretan bit : b1 b2 b3 .... bm
  - Operasi :
    - (b1 b2 b3...bm)$_{n-1}$ / Dn...D1 → sisa (R$_{n-1}$...R$_1$)
  - Dikirim b1 b2 b3...bm R$_{n-1}$...R$_1$

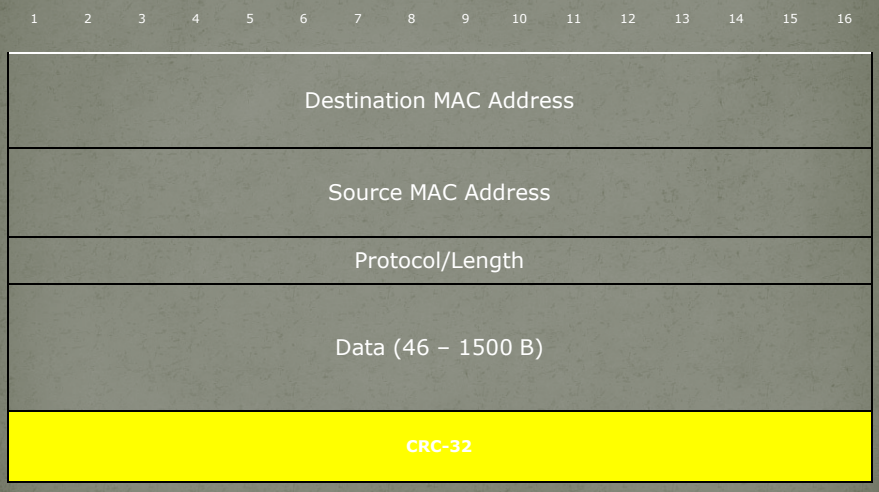
Cyclic Redudancy Check: Sisi Penerima

- Oleh penerima dilakukan operasi yang sama
  - b1 b2 b3…bm $R_{n-1}…R_1$ / $D_n…D_1$ → sisa ($r_{n-1}…r_1$)
  - Data benar jika $r_{n-1}…r_1 = 0$
  - Data salah jika $r_{n-1}…r_1 \neq 0$
- Pembagi standar internasional
  - CRC-16 → 11000000000000101
  - CRC-ITU → 10001000000100001
  - CRC-32 → 100000100100000010001110110110111
- 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)

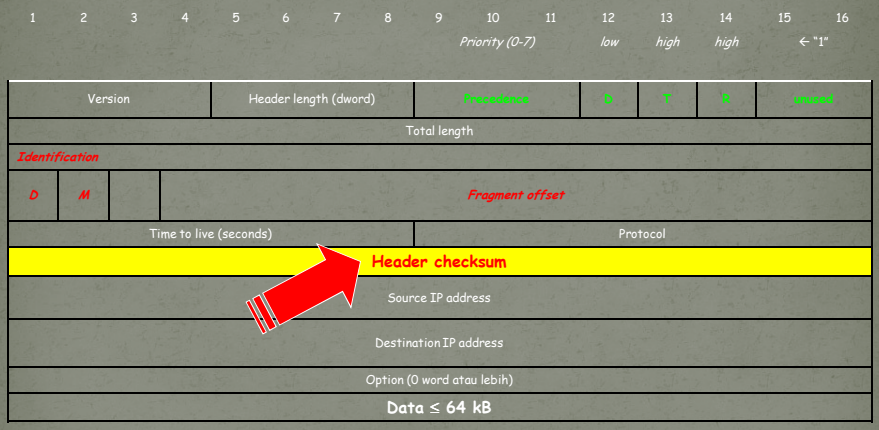


Checksum

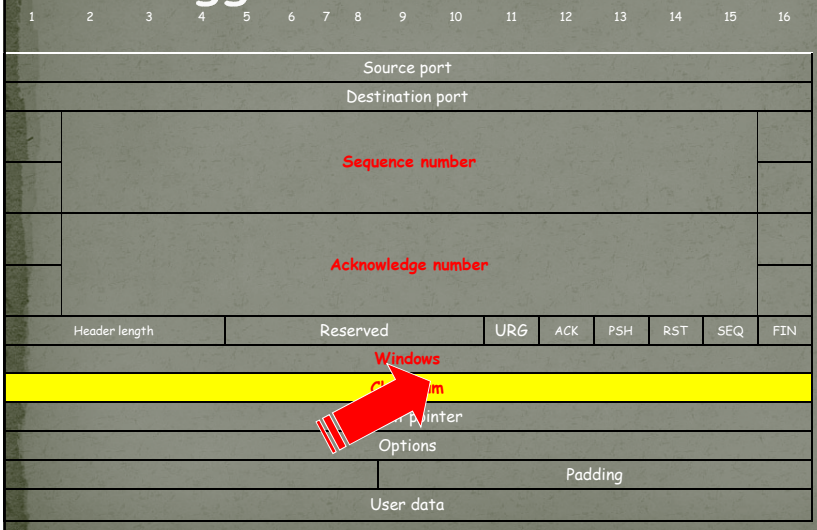
- CRC memerlukan perhitungan xor sebanyak jumlah bit data → memerlukan kemampuan komputasi yang cukup besar
- Diciptakan metoda checksum (untuk mengurangi perhitungan) pada beberapa jenis transmisi tidak perlu kecanggihan CRC atau sudah melakukan CRC di lapis lain
- Cara perhitungan checksum:
  - Data 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



Pengguna Checksum: TCP



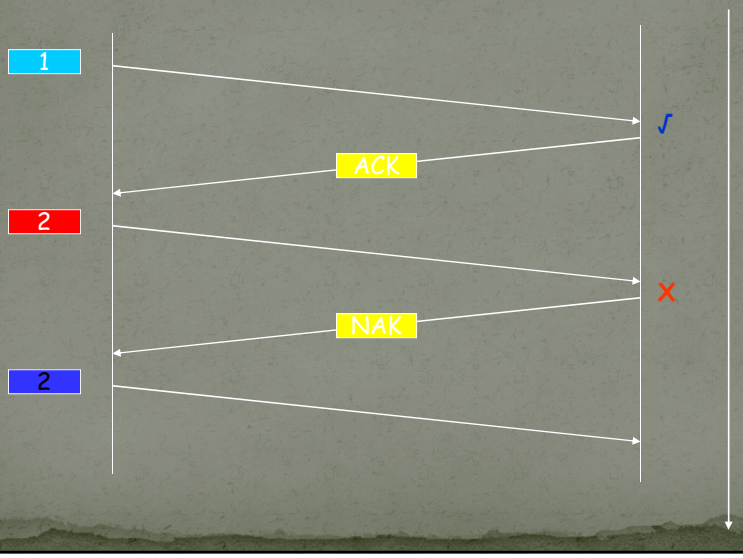
Backward Error Control

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

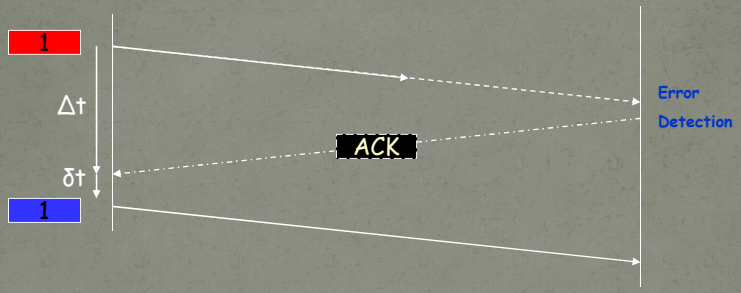
Backward Error Control: ARQ

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

ARQ : Idle RQ



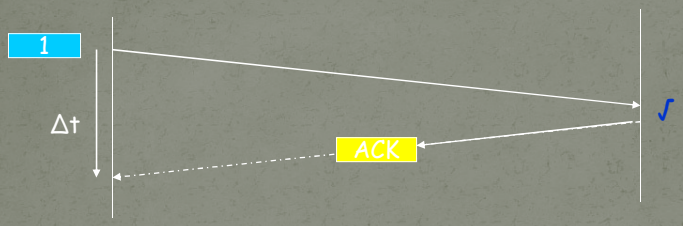
Kasus 1: jika paket tidak sampai



Pengirim menunggu feedback sampai Δt +δt, 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

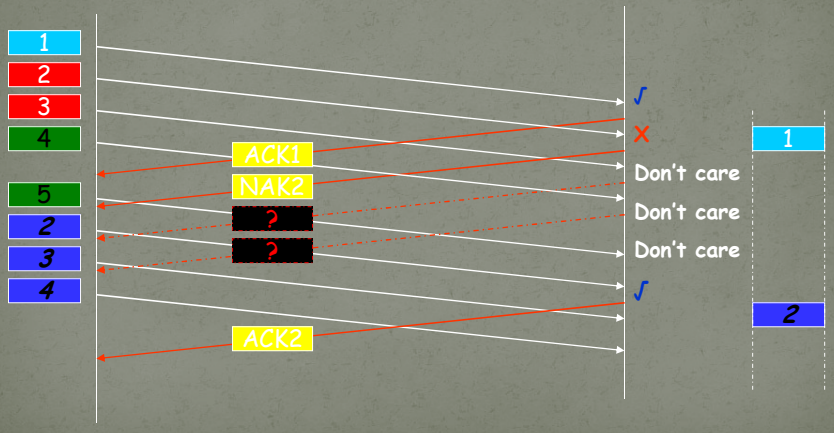
- "DIE HARD/Persistent/Ngotot" ARQ
- Paket akan diterima terjaga urutannya
- Efisiensi saluran paling rendah
- Cocok digunakan untuk saluran transmisi yang sangat jelek kualitasnya (banyak error)

ARQ : Selective Repeat

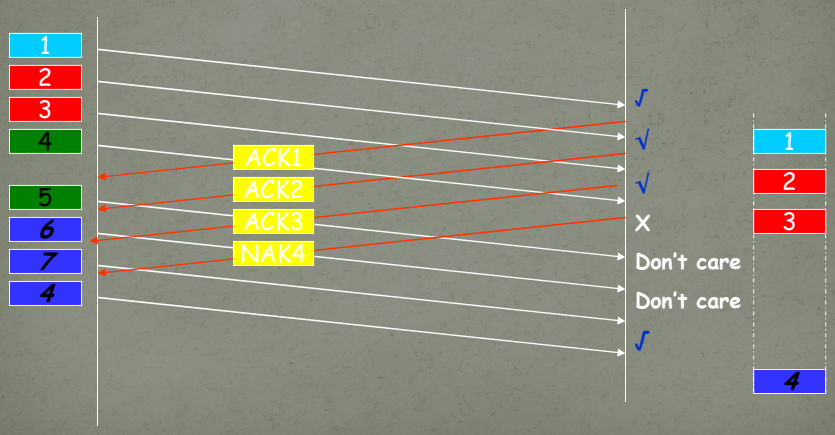
- Hanya mengirim ulang untuk paket yang salah
- Paket diterima tidak berurutan
- Efisiensi saluran tinggi (dibandingkan idle RQ)

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

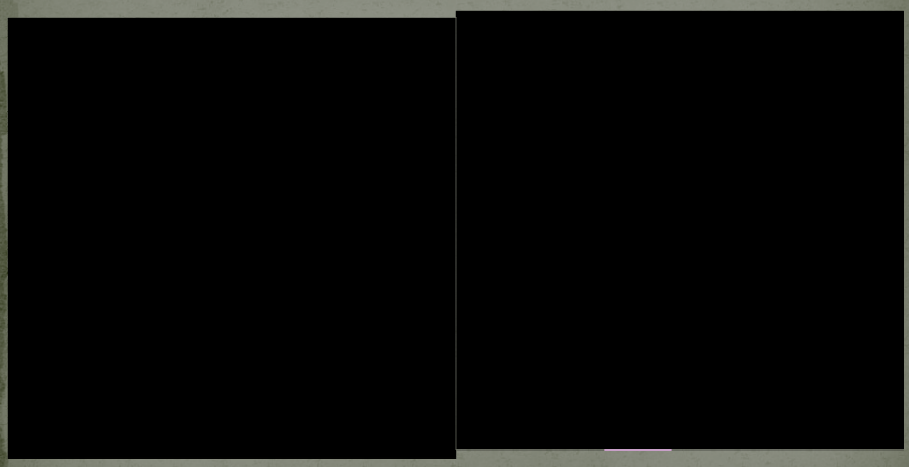
Forward Error Control

- 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

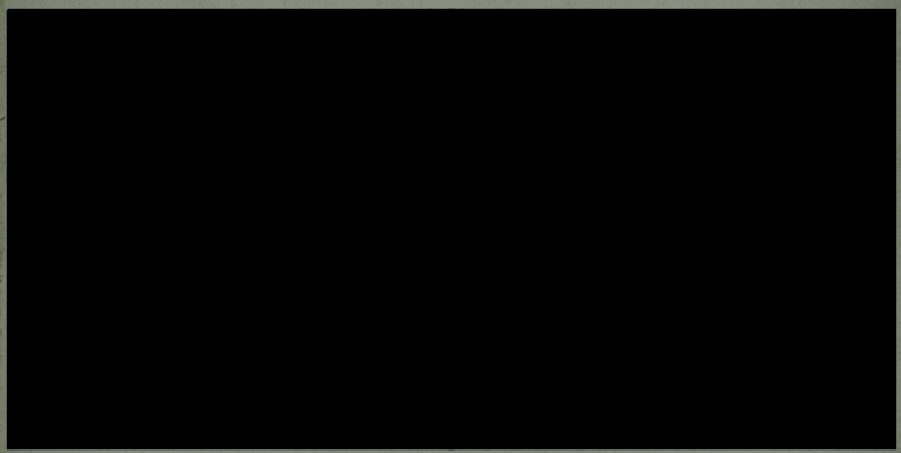
Jenis-Jenis FEC

- 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

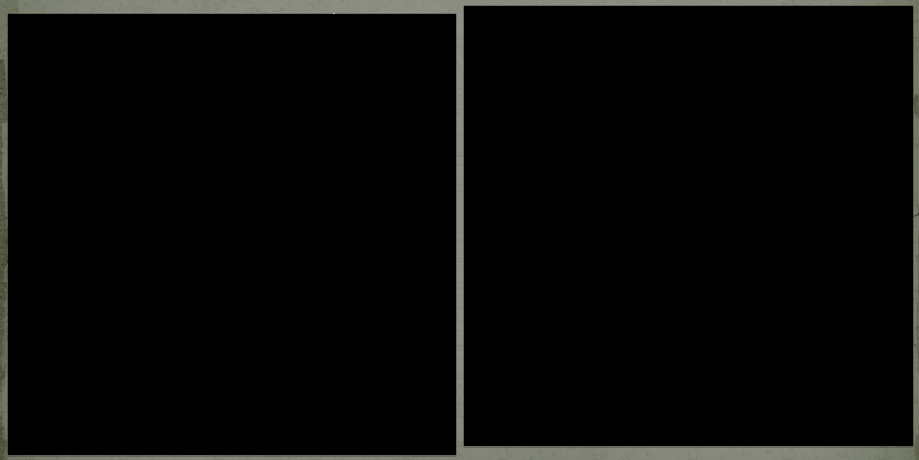
Contoh Block Parity 1



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:
  - P1 = d1 xor d2 xor d4 xor d5 xor d7 xor d9 dst
  - P2 = d1 xor d3 xor d4 xor d6 xor d7 xor d10 dst
  - P3 = d2 xor d3 xor d4 xor d8 xor d9 xor d10 dst
  - P4 = d5 xor d6 xor d7 xor d8 xor d9 xor d10 dst
  - P5 = d12 xor d13 xor d14 xor d15 dst
- Banyaknya bit pariti yang dibutuhkan tergantung jumlah bit datanya
- Sehingga deretan bit → P1 P2 d1 P3 d2 d3 d4 P4 d5 d6 d7 d8 d9 dst untuk ditransmisikan

0	0	0	1	P1
0	0	1	0	P2
0	0	1	1	D1
0	1	0	0	P3
0	1	0	1	D2
0	1	1	0	D3
0	1	1	1	D4
1	0	0	0	P4
1	0	0	1	D5
1	0	1	0	D6
1	0	1	1	D7
1	1	0	0	D8
1	1	0	1	D9
1	1	1	0	D10

Machine Way

P1 = d1 xor d2 xor d4 xor d5 xor d7 dst
P2 = d1 xor d3 xor d4 xor d6 xor d7 dst
P3 = ...
P4 = ...

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	
P4	P3	P2	P1	

- 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 xor d1 xor d2 xor d4 xor d5 xor d7 xor d9 dst
  - H2 = P2 xor d1 xor d3 xor d4 xor d6 xor d7 xor d10 dst
  - H3 = P3 xor d2 xor d3 xor d4 xor d8 xor d9 xor d10 dst
  - H4 = P4 xor d5 xor d6 xor d7 xor d8 xor d9 xor d10 dst
  - H5 = P5 xor d12 xor d13 xor d14 xor d15 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

- 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1
- Karena H4H3H2H1 = oooo → 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

- 1111111110101111
- Karena H4H3H2H1 = 0110 (6) → yang salah bit ke 6
- 111111**1**1101011 1
  → 1111011010111

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	
H4	H3	H2	H1	

Human Way

- 1111001110101111
- Karena H4H3H2H1 = 0101 (5) → yang salah bit ke 5
- 1111**0**01110101111

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	
H4	H3	H2	H1	

Human Way

1 1 1 1 0 0 0 1 1 0 1 0 1 1 1

- Karena H4H3H2H1 = 0010 → bit 2 salah
- Katanya yang benar adalah

1 0 1 1 0 0 0 1 1 0 1 0 1 1 1

- 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	
H4	H3	H2	H1	

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

D1	D2	D3	D4		P1	P2	D1	P3	D2	D3	D4
0	0	0	0		0	0	0	0	0	0	0
0	0	0	1		1	1	0	1	0	0	1
0	0	1	0		0	1	0	1	0	1	0
0	0	1	1		1	0	0	0	0	1	1
0	1	0	0		1	0	0	1	1	0	0
0	1	0	1		0	1	0	0	1	0	1
0	1	1	0		1	1	0	0	1	1	0
0	1	1	1		0	0	0	1	1	1	1
1	0	0	0		1	1	1	0	0	0	0
1	0	0	1		0	0	1	1	0	0	1
1	0	1	0		1	0	1	1	0	1	0
1	0	1	1		0	1	1	0	0	1	1
1	1	0	0		0	1	1	1	1	0	0
1	1	0	1		1	0	1	0	1	0	1
1	1	1	0		0	0	1	0	1	1	0
1	1	1	1		1	1	1	1	1	1	1

- Dmin = 3
- Error detection = 3-1
- Error correction = 3/2 = 1

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