Teknik Komunikasi Data Digital

Sinkronisasi :

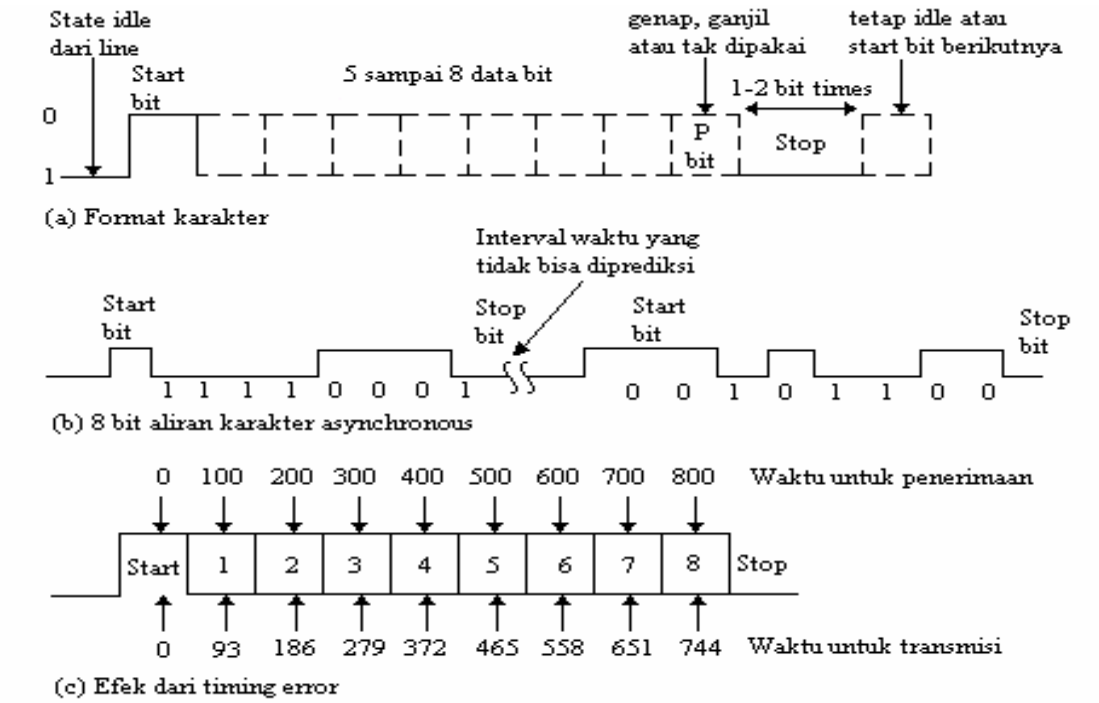
- Adalah satu kunci kerja dari komunikasi data.
- Transmiter mengirimkan pesan 1 bit pada satu saat melalui medium ke receiver.
- Receiver harus menandai awal dan akhir blok dari bit, juga harus diketahui durasi untuk masing-masing bit sehingga dapat sample lajur dari timing untuk membaca masing-masing bit (merupakan tugas dari timming).
- Contoh : jika ada perbedaan misalkan 1 % (clock receiver 1% lebih lambat atau lebih cepat daripada clock transmitter), maka pada pensamplingan pertama akan meleset dari tengah bit dan setelah jumlah waktu tertentu, akan mengalami error.

Sinkronisasi :

1. Asynchronous
   Untuk mencegah problem timming dengan tidak mengirim aliran bit panjang yang tidak putus-putusnya. Bit-bit dikirim per-karakter pada setiap waktu yang mana masing-masing karakter mempunyai panjang 5-8 bit. Timing atau synchronisasi harus dipertahankan antara tiap karakter; receiver mempunyai kesempatan untuk men-synchron-kan awal dari tiap karakter baru.

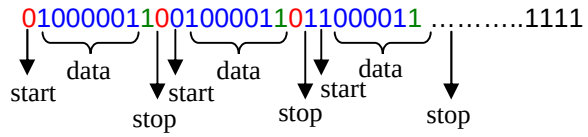
Keterangan gambar 4.1 :

- Idle (biasanya =’1’) jika tidak ada karakter yang ditransmisikan dan start bit = “0”, sedangkan jumlah karakter yang ditransmisikan antara 5-8 bit.
- Bit paritas digunakan untuk mendeteksi error, diatur oleh pengirim agar jumlah total ‘1’ termasuk bit paritas adalah genap, dan stop bit = ‘1’, yang panjangnya 1; 1,5; 2 kali durasi bit pada umumnya
- Komunikasi asinkron adalah sederhana dan murah, tetapi memerlukan overhead dari 2 ke 3 bit per karakter, prosentasi overhead dapat dikurangi dengan mengirimkan blok-blok bit besar antara bit start dan bit stop
- Contoh : akan dikirimkan data ASCII ABC dengan A = 41H, B = 42H dan C = 43H tanpa paritas, maka :
  A = $0100\ 0001_2$ invert kode ASCII 7 bit $100\ 0001_2$
  B = $0100\ 0010_2$ invert kode ASCII 7 bit $010\ 0001_2$
  C = $0100\ 0011_2$ invert kode ASCII 7 bit $110\ 0001_2$



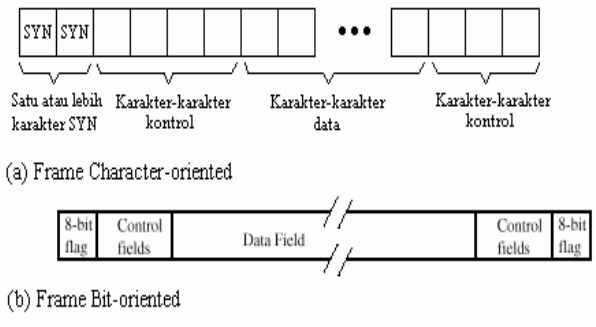
Gambar 4.1. Transmisi Asynchronous

Datanya terdiri dari kode ASCII 7 bit dan bit stop, maka :



Kode 7 bit memakai panjang 1 bit untuk bit start dan 1 bit untuk bit stop, maka overheadnya 2 / 9 = 0,22.

2. Synchronous / timing



- ♥ Efisien, karena blok-blok karakter / bit-bit ditransmisikan tanpa kode start dan stop, tetapi tiap blok blok dimulai dengan suatu pola *preamble* bit dan diakhiri dengan pola *postamble* bit. Pola-pola ini adalah kontrol informasi.
- ♥ Waktu kedatangan dan keberangkatan untuk masing-masing bit dapat diramalkan.
- ♥ **Frame** adalah data plus kontrol informasi. Format framenya tergantung dari metode transmisi, yaitu
  - Transmisi orientasi karakter
    - Blok-blok data dikerjakan sebagai barisan karakter (biasanya 8 bit karakter), frame dimulai dengan 1 atau lebih karakter sinkronisasi. Karakter sinkronisasi biasanya disebut dengan "SYN" yang merupakan bit pattern unik sinyal yang diterima penerima permulaan dari blok.
    - Penerima kemudian merubah blok-blok data yang datang oleh karakter SYN dan menerima data sampai karakter postamble (informasi yang terletak pada bagian belakang blok data yang dikirimkan) terlihat dan begitu seterusnya
  - Transmisi bit.
    - Blok-blok data dikerjakan sebagai barisan bit-bit, tidak ada data maupun informasi kontrol diperlukan untuk menginter-prestasikan dalam satuan karakter 8 bit
    - Pada awal terdapat flag, begitu juga pada akhir yang panjangnya 8 bit yang berguna sebagai awal dan akhir untuk penerima

Perbandingan asinkron dan sinkron

- ♥ Untuk blok-blok data yang cukup besar, transmisi sinkronisasi jauh lebih efisien daripada asinkron. Transmisi asinkron memerlukan overhead 20 % atau lebih.
- ♥ Bila menggunakan transmisi sinkron biasanya lebih kecil dari 1000 bit, yang mengandung 48 bit kontrol informasi (termasuk flag), maka untuk pesan 1000 bit, overheadnya adalah 48 / 1048 X 100% = 4.6%

Urutan pengerjaan sinkronisasi yaitu :

1. Sinkronisasi bit
   Ditandai awal & akhir untuk masing-masing bit
2. Sinkronisasi karakter / kata
   Ditandai awal dan akhir untuk masing-masing karakter / satuan kecil lainnya dari data
3. Sinkronisasi blok / pesan
   Ditandai awal dan akhir dari satuan besar data. Dan untuk pesan yang besar, dibagi-bagi menjadi beberapa blok kemudian baru dikirimkan pengurutan blok-blok yang telah dibagi tersebut adalah tugas dari timming. Sedangkan pengaturan level sinyal adalah tugas dari sintax dan untuk melihat arti dari pesan adalah tugas dari semantik.

Deteksi error dengan Redundansi, yaitu data tambahan yang tidak ada hubungannya dengan isi informasi yang dikirimkan, berupa bit pariti. Berfungsi menunjukkan ada tidaknya kesalahan data. Yaitu dengan mendeteksi dan mengoreksi kesalahan yang terjadi. Makin banyak redundansi makin baik deteksi errornya. Akibatnya makin rendah troughput dari data yang berguna.

Troughput adalah perbandingan antara data yang berguna dengan data keseluruhan. Banyaknya tambahan pada redundansi sampai 100% dari jumlah bit data.

Teknik mendeteksi error :
Teknik deteksi error menggunakan **error-detecting-code**, yaitu tambahan bit yang ditambah oleh transmitter. Dihitung sebagai suatu fungsi dari transmisi bit-bit lain. Pada receiver dilakukan perhitungan yang sama dan membandingkan kedua hasil tersebut, dan bila tidak cocok maka berarti terjadi deteksi error. Dan Apabila sebuah frame ditransmisikan ada 3 kemungkinan klas yang dapat didefinisikan pada penerima, yaitu :

1. Klas 1 (P_1) : Sebuah frame datang dengan tidak ada bit error (*jadi tidak berarti dalam mendeteksi error, karena nggak ada error!*)
2. Klas 2 (P_2) : Sebuah frame datang dengan 1 atau lebih bit error yang tidak terdeteksi
3. Klas 3 (P_3) : Sebuah frame datang dengan 1 atau lebih bit error yang terdeteksi dan tidak ada bit error yang tidak terdeteksi. (*nggak berarti juga, semua error udah terdeteksi*)

Ada dua pendekatan untuk deteksi kesalahan :

1. **Forward Error Control**
   Dimana setiap karakter yang ditransmisikan atau frame berisi informasi tambahan (redundant) sehingga bila penerima tidak hanya dapat mendeteksi dimana error terjadi, tetapi juga menjelaskan dimana aliran bit yang diterima error.
2. **Feedback (backward) Error Control**
   Dimana setiap karakter atau frame memilki informasi yang cukup untuk memperbolehkan penerima mendeteksi bila menemukan kesalahan tetapi tidak lokasinya. Sebuah transmisi kontro digunakan untuk meminta pengiriman ulang, menyalin informasi yang dikirimkan.

Feedback error control dibagi menjadi 2 bagian, yaitu :

1. Teknik yang digunakan untuk deteksi kesalahan
2. Kontrol algoritma yang telah disediakan untuk mengontrol transmisi ulang.

Metode Deteksi Kesalahan :

1. *Echo*
   Metode sederhana dengan sistem interaktif. Operator memasukkan data melalui terminal dan mengirimkan ke komputer. Komputer akan menampilkan kembali ke terminal, sehingga dapat memeriksa apakah data yang dikirimkan dengan benar.
2. *Error Otomatis / Parity Check*
   Penambahan parity bit untuk akhir masing-masing kata dalam frame. Tetapi problem dari parity bit adalah impulse noise yang cukup panjang merusak lebih dari satu bit, pada data rate yang tinggi.

   Jenis Parity Check :
   a. Even parity (paritas genap), digunakan untuk transmisi asynchronous. Bit parity ditambahkan supaya banyaknya '1' untuk tiap karakter / data adalah genap
   b. Odd parity (paritas ganjil), digunakan untuk transmisi synchronous. Bit parity ditambahkan supaya banyaknya '1' untuk tiap karakter / data adalah ganjil

Dengan bit pariti dikenal 3 deteksi kesalahan, yaitu :

a. **Vertical Redundancy Check / VRC**
   Setiap karakter yang dikirimkan (7 bit) diberi 1 bit pariti. Bit pariti ini diperiksa oleh penerima untuk mengetahui apakah karakter yang dikirim benar atau salah. Cara ini hanya dapat melacak 1 bit dan berguna melacak kesalahan yang terjadi pada pengiriman berkecepatan menengah, karena kecepatan tinggi lebih besar kemungkinan terjadi kesalahan banyak bit. Kekurangan : bila ada 2 bit yang terganggu ia tidak dapat melacaknya karena paritinya akan benar.
   Contoh :
   ASCII huruf "A" adalah 41h
   100 0001 ASCII 7 bit
   1100 0001 ASCII dengan pariti ganjil
   0100 0001 ASCII dengan pariti genap
   Akibatnya huruf "A" kode ASCII dalam Hex :
   - 41 bilamana pariti genap
   - A1 bilamana pariti ganjil

b. **Longitudinal Redundancy Check / LRC**
   LRC untuk data dikirim secara blok. Cara ini seperti VRC hanya saja penambahan bit pariti tidak saja pada akhir karakter tetapi juga pada akhir setiap blok karakter yang dikirimkan. Untuk setiap bit dari seluruh blok karakter ditambahkan 1 bit pariti termasuk juga bit pariti dari masing-masing karakter. Tiap blok mempunyai satu karakter khusus yang disebut Block Check Character (BCC) yang dibentuk dari bit uji. dan dibangkitkan dengan cara sebagai berikut :
   "Tiap bit BCC merupakan pariti dari semua bit dari blok yang mempunyai nomor bit yang sama. Jadi bit 1 dari BCC merupakan pariti genap dari semua bit 1 karakter yang ada pada blok tersebut, dan seterusnya"
   Kerugian : terjadi overhead akibat penambahan bit pariti per 7 bit untuk karakter.

c. **Cyclic Redundancy Check / CRC**
   Digunakan pengiriman berkecepatan tinggi, sehingga perlu rangkaian elektronik yang sukar. Cara CRC mengatasi masalah overhead dan disebut pengujian berorientasi bit, karena dasar pemeriksaan kemungkinan kesalahan adalah bit / karakter dan menggunakan rumus matematika khusus.

Contoh menggunakan paritas genap :

							VRC	
1	0	1	1	0	1	1	***1***	'1'=5
1	1	0	1	0	1	1	***1***	'1'=5
0	0	1	1	1	0	1	***0***	'1'=4
1	1	1	1	0	0	0	***0***	'1'=4
1	0	0	0	1	0	1	***1***	'1'=3
0	1	0	1	1	1	1	***1***	'1'=5
0	***1***	***1***	***1***	***1***	***1***	***1***		***LRC***
'1'=4	'1'=3	'1'=3	'1'=5	'1'=3	'1'=3	'1'=5		

Satu blok informasi dilihat sebagai sederetan bit yang ditransmisikan. Bit yang ditransmisikan dimasukkan kedalam register geser siklis yang disebut generator CRC. Operasi ini didasarkan atas pembagian deretan bit dengan sebuah fungsi khusus.
Hasil bagi pembagian diabaikan. Sisanya disalurkan sebagai BCS (Block Check Sequence). Fungsi khusus tersebut disebut generator polynominal.

Realisasi Generator CRC / Penguji
Data dimasukkan kedalam register geser pada berbagai titik melalui gerbang XOR yang mempunyai hubungan langsung dengan generator polynominal yaitu rumus matematika untuk membagi bit data. Lebih baik dari VRC / LRC, 99% error dapat terdeteksi.

Contoh hasil pada register geser 16 bit dengan gerbang XOR, dimana input pada bit 0, 5, 12 dan output pada bit 15, maka :

- CRC-CCITTT $= X^{16} + X^{12} + X^5 + 1$
- CRC-16 $= X^{16} + X^{15} + X^2 + 1$
- CRC-12 $= X^{12} + X^{11} + X^3 + X^2 + 1$
- LRC $= X^8 + 1$
- RC-32 $= X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$

3. *Framing Check*
   Dipakai pada transmisi asinkron dengan adanya bit awal dan akhir. Data berada diantara bit awal dan bit akhir. Dengan memeriksa kedua bit ini dapat diketahui apakah data dapat diterima dengan baik atau tidak. Transmisi asinkron mempunyai bentuk bingkai sesuai dengan ketentuan yang dipergunakan

Pendekatan yang umum dipakai adalah data link layer memecah aliran bit menjadi frame-frame diskrit dan menghitung checksum setiap framenya. Ketika sebuah frame tiba di tujuan, checksum dihitung kembali. Bila hasil perhitungan ulang checksum tersebut berbeda dengan yang terdapat pada frame, maka data link layer akan mengetahui bahwa telah terjadi error dan segera akan mengambil langkah tertentu sehubungan dengan adanya error tersebut (misalnya, membuang frame yang buruk dan mengirimkan kembali laporan error).

Salah satu cara untuk melaksanakan pembuatan frame ini adalah dengan cara menyisipakn gap waktu di antara dua buah frame, sangat mirip seperti spasi antara dua buah katan dalam suatu teks. Akan tetapi, jaringan jarang memberikan jaminan tentang pewaktuan. Karena itu, mungkin saja gap ini dibuang, atau diisi oleh gap lainnya selama proses transmisi, karena sangat besar risikonya dalam menghitung pewaktuan untuk menandai awal dan akhir frame, telah dibuat metode lainnya, yaitu 4 buah metoda :

1. Karakter penghitung
2. Pemberian karakter awak dan akhir, dengan pengisian karakter
3. Pemberian flag awal dan akhir, dengan pengisian bit
4. Pelanggaran pengkodean physical layer.

Metoda framing pertama menggunakan sebuah field pada header untuk menspesifikasikan jumlah karakter di dalam frame. Ketika data link layer pada mesin yang dituju melihat karakter penghitung, maka data link layer akan mengetahui jumlah karakter yang mengikutinya, dan kemudian juga akan mengetahui posisi ujung frame-nya. Masalah yang dijumpai dalama algoritma ini adalah bahwa hitungan dapat dikacaukan oleh error transmisi. Misal

bila hitungan karakter 5 frame menjadi 7, maka tempat yang dituju akan tidak sinkron dan tidak dapat mengetahui awal frame berikutnya.

Bahkan bila cheksum tidak benar sehingga tempat yang dituju mengetahui bahwa frame yang bersangkutan buruk, maka tidak mungkin untuk menentukan awal frame berikutnya. Pengiriman kembali sebuah frame ke sumber untuk meminta pengiriman ulangpun tidak akan menolong, karena tempat yang dituju tidak mengetahui jumlah karakter yang terlewat untuk mendapatkan awal transmisi. Untuk alasan ini, metoda hitungan karakter ini sudah jarang digunakan lagi.

Metode framing yang kedua mengatasi masalah resinkronisasi setelah terjadi suatu error dengan membuat masing-masing frame diawali dengan sederetan karakter DLE STX ASCII dan diakhiri dengan DLE ETX (DLE=Data Link Escape, STX=Start Of Text, ETX=End Of Text). Dalam metoda ini, bila tempat yang dituju kehilangan track batas-batas frame, maka yang perlu dilakukan adalah mencari karakter-karakter DLE STX dan DLE ETX. Masalah serius yang terjadi pada metoda ini adalah ketika data biner, seperti program object, atau bilangan floating point, ditransmisikan. Karakter-karakter DLE STX dan DLE ETX yang terdapat pada data mudah sekali menganggu framing. Satu cara untuk mengatasi masalah ini adalah dengan membuat data link pengirim menyisipkan sebuah karakter DLE ASCII tepat sebelum karakter DLE "insidentil" pada data. Data link layer pada mesin penerima membuang DLE sebelum data diberikan ke network layer. Teknik ini disebut character stuffing (pengisian karakter). DLE-DLE pada data selalu digandakan. Kerugian penting dalam memakai metoda framing ini sangat berkaitan erat dengan karakter 8-bit secara umum dan kode karakter ASCII pada khususnya. Dengan berkembangnya jaringan, kerugian dari melekatkan kode karakter dalam mekanisma framing menjadi semakin jelas, sehingga suatu teknik baru perlu dibuat untuk memungkinkan pemakaian karakter berukuran sembarang.

Teknik baru memungkinkan frame data berisi sembarang sejumlah bit dan mengijinkan kode karakter dengan sembarang jumlah bit per karakter. Teknik ini bekerja seperti berikut, setiap frame diawali dan diakhiri oleh pola bit khusus, 01111110 yang disebut flag. Kapanpun data link layer pada pengirim menemukan lima buah flag yang berurutan pada data, maka data link layer secara otomatis mengisikan sebuah bit 0 ke aliran bit keluar. Pengisian bit ini analog dengan pengisian karakter, dimana sebuah DLE diisikan ke aliran karakter keluar sebelum DLE pada data. Ketika penerima melihat 5 buah bit 1 masuk yang berurutan, yang diikuti oleh sebuah bit 0, maka penerima secara otomatis mengosongkan (menghapus) bit 0 tersebut. Seperti halnya pengisian karakter transparan sepenuhnya bagi network layer pada kedua buah komputer, demikian pula halnya dengan pengisian bit. Bila data pengguna berisi pola flag 01111110, maka flag ini akan ditransmisikan kembali sebagai 011111010 tapi akan disimpan di memory penerima sebagai 01111110. Dengan pengisian bit, maka batas antara dua frame dapat dikenal jelas oleh pola flag. Jadi bila penerima mengalami kehilangan track frame tertentu, yang perlu dilakukan adalah menyisir input deretan flag, karena flag tersebut hanya mungkin terdapat pada batas frame saja dan tidak pernah berada pada data.

Metode framing terakhir hanya bisa digunakan bagi jaringan yang encoding pada medium fisiknya mengandung beberapa redundansi (pengulangan). Misalnya, sebagian LAN melakukan encode bit 1 data dengan menggunakan 2 bit fisik. Umumnya, bit1 merupakan pasangan tinggi-rendah dan bit 0 adalah pasangan rendah-tinggi. Kombinasi pasangan tinggi-tinggi dan rendah-rendah tidak digunakan bagi data. Proses itu berarti bahwa setiap bit data memiliki transisi di tengah, yang memudahkan penerima untuk mencari batas bit. Manfaat kode fisik yang invalid merupakan bagian standard LAN 802.2.