B A B VIII

DATA LINK CONTROL

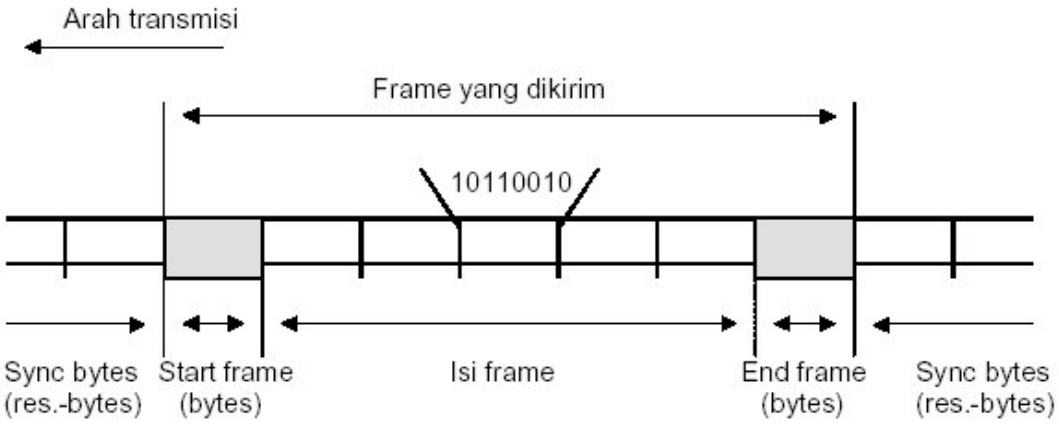
Agar komunikasi data digital berlangsung efektif, banyak hal yang akan diperlukan untuk mengontrol dan mengatur pertukaran data. Agar sistem pengontrolan yang diperlukan dapat tercapai diperlukan layer yang secara logika ditambahkan diatas *physical-interface*, logika yang ditambahkan tersebut dinamakan sebagai *data-link-control* atau *data-link-control-protocol.* Untuk melihat kegunaan *data-link-control,* maka ditampilkan beberapa hal yang berkaitan dengan komunikasi data agar berjalan efektif diantara dua *station* (transmiter-receiver) yang terhubung, yang meliputi :

- *Frame-synchronization*
  Data dikirimkan dalam bentuk blok yang disebut frame, awal dan akhir masingmasing frame harus dapat dikenali.
- *Flow-control*
  Station pengirim tidak akan mengirim frame pada kecepatan yang tinggi jika station penerima tidak dapat menangkapnya.
- *Error-control*
  Beberapa bit error yang dikenali dalam sistem transmisi harus dapat diperbaiki/betulkan.
- *Addressing*
  Pada lintasan yang bertitik banyak, seperti misalnya pada *local-area-network*, maka identitas dari kedua station yang terlibat didalam transmisi harus spesifik (diperinci).
- *Control* dan Data pada link yang sama
  Tidak diperlukan sekali untuk memiliki jalur komunikasi yang terpisah secara fisik untuk informasi pengontrol, tetapi penerima (receiver) harus dapat membedakan informasi pengontrol dari data yang sedang dikirimkan.
- *Link-management*
  Untuk memulai, merawat dan memutus lintasan komunikasi yang menopang pertukaran data membutuhkan sejumlah koordinasi dan kerjasama beberapa station, sehingga diburuhkan suatu prosedur untuk mengatur pertukaran data ini.

7.1. TRANSMISI SINKRON

Untuk transmisi data dengan blok data yang besar dan kecepatan transfer yang tinggi, maka dapat digunakan transmisi sinkron sebagai alternatif. Dengan transmisi sinkron blok (frame) dari data secara lengkap dikirimkan seperti deretan bit yang berdekatan tanpa adanya delay diantara elemen karakter, tidak seperti transmisi asinkron yang ditandai dengan start-bit dan stop-bit tiap elemen karakter 7 bit.

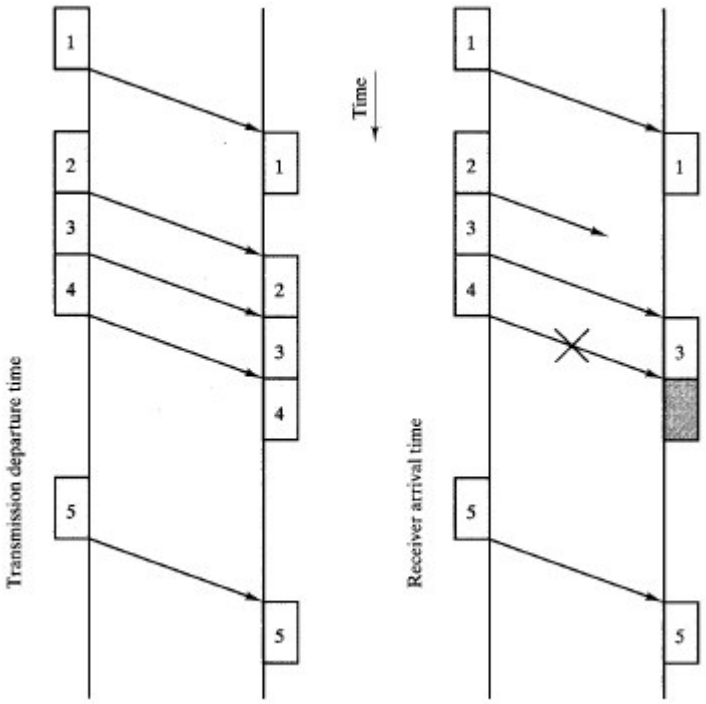
- Deretan bit yang dikirimkan memiliki pengkodean yang sama antara transmitter dan receiver sehingga receiver dapat memperoleh data secara sinkron.
- Semua frame yang kan dikirim terlebih dahulu diawali oleh *reserved-bytes* untuk memastikan receiver siap untuk menterjemahkan deretan bit agar didapatkan data yang benar.
- Isi dari masing-masing frame terbungkus oleh sepasang *reserved-bytes* sebagai sinkronisasi frame.



Gambar 7.1 *Sinyal Transmisi Sinkron*

7.2. FLOW CONTROL

Flow-control adalah suatu teknik untuk menjamin bahwa entitas pengirim tidak akan membanjiri data kepada entitas penerima. Entitas penerima secara khusus mengalokasikan *buffer* dengan beberapa kali panjangnya tansfer. Ketika data diterima receiver harus mengerjakan sejumlah proses tertentu sebelum mengalirkan data ke software dengan level yang lebih tinggi. Dengan tidak adanya *flow-control* maka *buffer* pada penerima dapat terisi penuh dan melebihi kapasitas, bersamaan pada saat penerima masih memproses data sebelumnya. Sebagai permulaannya maka kita menguji mekanisme *flow-control* dengan tidak adanya error, seperti ditunjukkan pada gambar 7.2 dibawah. Sumbu keatas adalah urutan waktu yang akan mempermudah dalam mengambarkan hubungan kirimdan terima yang benar sebagai fungsi waktu. Masing-masing tanda panah menunjukkan satu frame data yang sedang *transit* (dalam pejalanan) diantara dua station. Data dikirimkan dalam urutan frame yang masing-masing frame berisi bagian data dan sejumlah informasi pengontrol.



Gambar 7.2 *Model Transmisi Frame*

Diasumsikan bahwa semua frame yang dikirimkan berhasil diterima dengan sukses, tidak ada frame yang hilang dan tidak ada frame yang datang mengalami error. Selanjutnya frame-frame tersebut tiba bersamaan dengan dikirimkannya frame, bagaimanapun juga masing-masing frame yang dikirimkan sebelum diterima akan mendapat *delay* pasa saluran yang besarnya berubah-ubah.

Stop-And-Wait Flow-Control

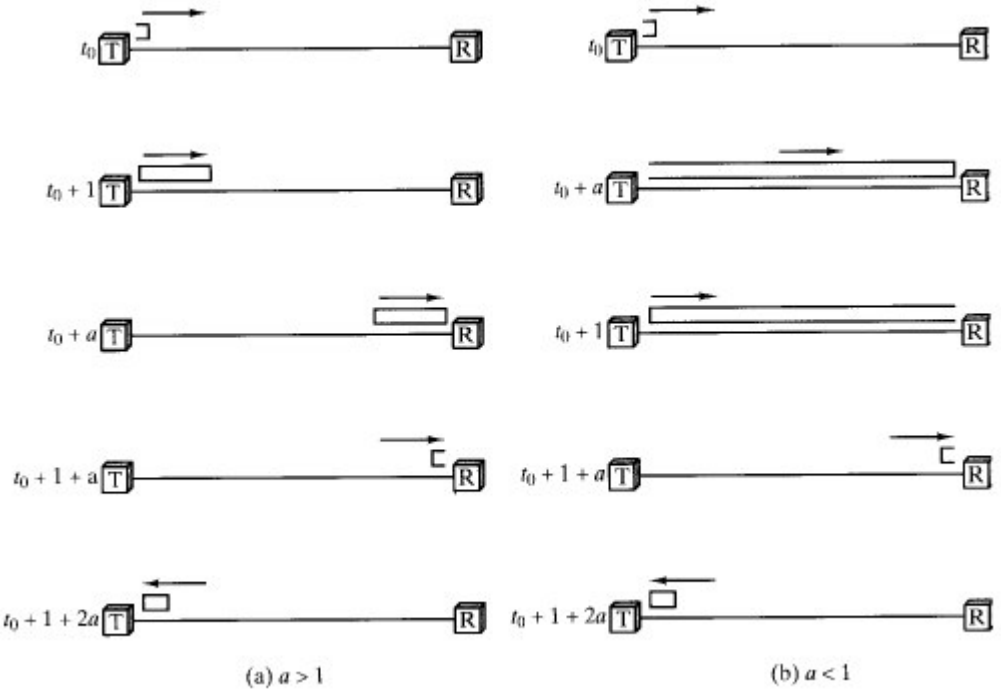
Bentuk sederhana dari *flow-control* adalah *stop-and-wait flow-control* yang bekerja sebagai berikut. entitas sumber mengirimkan frame, setelah diterima entitas tujuan memberi tanda untuk menerima frafe berikutnya dengan mengirimkan balasan sesuai frame yang telah diterima. entitas sumber harus menunggu sampai ia menerima balasan dari entitas tujuan sebelum mengirimkan frame berikutnya. Selanjutnya entitas sumber dapat menghentikan aliran data dengan menahan jawaban. Prosedur ini dapat bekerja dengan baik tentunya bila data dikirimkan dalam jumlah frame yang besar, dalam hal ini entitas sumber akan membagi blok data yang banyak menjadi blok data yang lebih kecil yang kemudian dikirimkan dalam beberapa frame.

Stop-and-wait digunakan untuk transmisi dengan keperluan tertentu, yang memiliki beberapa ciri-ciri :

- Ukuran *buffer* pada receiver terbatas.
- Transmisi dapat lebih banyak, sebab bila dikirimkan secara langsung sebanyak frame dari data yang ada, maka akan mudah menimbulkan error, sehingga dengan frame yang lebih kecil maka error dapat dideteksi lebih awal dari sejumlah frame data yang dikirimkan.

- Pada pemakaian bersama sebuah media atransmisi (misalkan LAN), umumnya tidak diijinkan untuk menempati media transmisi dalam waktu yang lama yang menyebabkan *delay* pada *station* lain yang akan melakukan transmisi.

Pada gambar 7.3 dibawah, berisi urutan gambar dari sebuah proses transmisi (dengan waktu transmisi = 1, dan waktu propagasi = a). Pada urutan 1 sampai dengan 4 ditunjukkan proses transmisi frame yang berisi data, dan pada gambar yang terakhir (urutan nomer 5) menunjukkan kembalinya frame jawaban yang kecil. Dengan catatan bahwa pada a>1 menjunjukkan media transmisi sedang sibuk (dipergunakan statiun lain), sedangkan a<1 media transmisi sedang kosong sehingga dapat dipergunakan secara effisien.

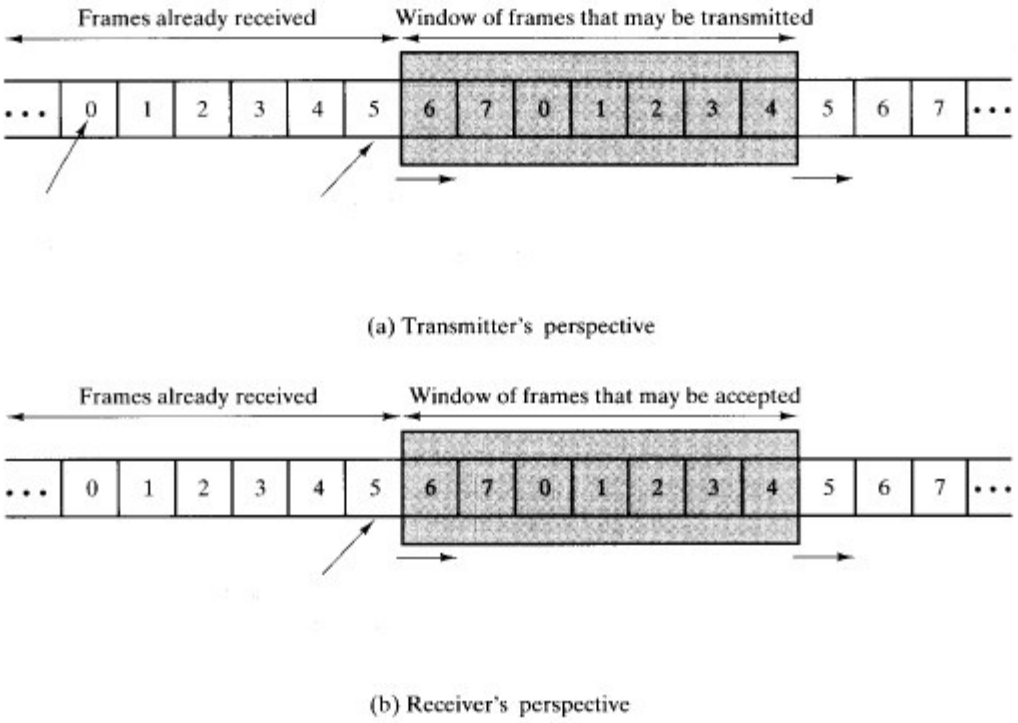


Gambar 7.3 *Pemanfaatan Saluran untuk Stop-And-Wait*

Sliding-Window Flow Control

Masalah utama yang selama ini adalah bahwa hanya satu frame yang dapat dikirimkan pada saat yang sama. Dalam keadaan antrian bit yang akan dikirimkan lebih besar dari panjang frame (a>1) maka diperlukan suatu efisiensi. Untuk memperbesar efisiensi yang dapat dilakukan dengan memperbolehkan transmisi lebih dari satu frame pada saat yang sama. Bila suatu *station A* dan *B* dihubungkan dengan jalur *full-duplex*, *station B* mengalokasikan buffers dengan selebar n frame, yang berarti stasiun B dapat menerima n frame, dan *station A* diperbolehkan untuk mengirim frame sebanyak n tanpa menunggu adanya jawaban. Untuk menjaga jejak dimana frame yang dikirimkan sedang dijawab maka masing-masing jawaban diberi label dengan nomor yang urut. *Station B* menjawab frame dengan mengirimkan jawaban yang dilengkapi nomor urut dari frame berikutnya yang diinginkan. Jawaban ini juga memiliki maksud untuk memberitahukan bahwa *station B* siap untuk menerima n frame berikutnya, dimulai dengan nomer urut yang telah tercantum. Skema ini juga dapat dipergunakan untuk menjawab lebih dari satu frame. Misalnya *station B* dapat menerima frame 2, 3 dan 4, tetapi menahan

jawaban sampai samapai frame ke 4 tiba, dengan kembali jawaban dengan nomer urut 5, *station B* menjawab frame 2, 3, dan 4 pada satu saat. Station A memeliharan daftar nomer urutan yang boleh dikirim, sedangkan *station B* memelihara daftar nomer urutan yang siap akan diterima. Masing-masing daftar tersebut dapat dianggap sebagai window dari frame, sehingga prinsip kerjanya disebut dengan pengontrol aliran *sliding-window.* Diperlukan untuk dibuat komentar tambahan untuk masing-masing, karena nomer urut yang dipakai menempati daerah didalam frame, komentar tambahan ini dibatasai oleh terbatasnya tempat yang tersedia. Misalnya untuk daerah dengan panjang 3 bit, maka nomer urut jangkauannya antara 0 s/d 7 saja, sehingga frame diberi nomer dengan modulo 7, jadi sesudah nomer urut 7 berikutnya adalah nomer 0. Pada umumnya untuk daerah dengan apnjang k-bit, maka jangkauan nomer urut dari 0 samapai dengan 2k-1, dan frame diberi nomer dengan modulo 2k. Pada gambar 7.4 dibawah menggambarkan proses *sliding-windows*, dengan diasumsikan nomer urut menggunakan 3-bit sehingga frame diberi nomer urut 0 s/d 7, selanjutnya nomer yang sama dipakai kembali sebagai bagian urutan frame. Gambar segiempat yang diberi bayangan (disebut *window*) menunjukkan transmitter dapat mengirimkan 7 frame, dimulai dengan frame nomer 7. Setiap waktu frame dikirimkan maka *window* yang digambarkan sebagai kotak dibayangi akan menyusut, setiap waktu jawaban diterima, *window* akan membesar. Ukuran panjang *window* sebenarnya tidak diperlukan sebanyak ukuran maksimumnya untuk diisi sepanjang nomer urut. Sebagai contoh, nomer urut menggunakan 3-bit, stasiun dapat membentuk *window* dengan ukuran 4, menggunakan protokol pengatur aliran *sliding-window*. Sebagai contoh diasumsikan memiliki daerah nomer urut 3-bit dan maksimum ukuran window adalah 7 frame. Dimulai dari *station A* dan *B* telah menandai *window* dan *station A* mengirimkan 7 frame yang dimulai dengan frame 0 (F0), sesudah mengirimkan 3 frame (F0, F1, dan F2) tanpa jawaban maka *station A* telah menyusutkan *window* nya menjadi 4 frame. *Window* menandati bahwa *station A* dapat mengirimkan 4 frame, dimulai dari frame nomer 3 selanjutnya *stasiun B* mengirim receive-ready (RR) yang berarti semua frame telah diterima sampai frame nomer 2 dan selanjutnya siap menerima frame nomer 3, tetapi pada kenyataannya disiapkan menerima 7 frame, dimulai frame nomer 3. *Station A* terus mengirimkan frame nomer 3, 4, 5, dan 7, kemudian *station B* menjawab RR7 sebagai jawaban dari semua frame yang diterima dan pengusulkan *station A* mengirim 7 frame, dimulai frame nomer 7.

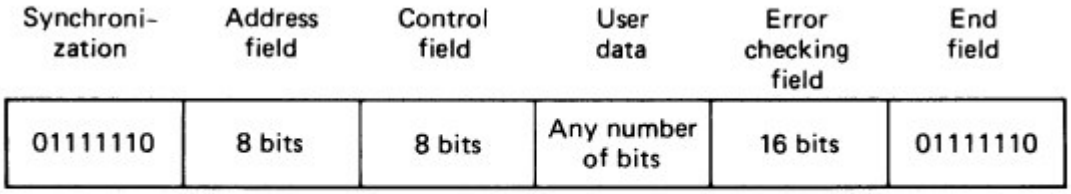


Gambar 7.4 *Skema Aliran Sliding-Window*

Receiver harus dapat menampung 7 frame belebihi satu jawaban yang telah dikirim, sebagian besar protokol juga memperbolehkan suatu station untuk memutuskan aliran frame dari sisi (arah) lain dengan cara mengirimkan pesar *receive-not-ready* (RNR), yang dijawab frame terlebih dulu, tetapi melarang transfer frame berikutnya. Bila dua stasiun saling bertukar data (dua arah) maka masing-masing perlu mengatur dua *window*, jadi satu untuk transmit dan satu untuk receive dan masingmasing sisi (arah) saling mengirim jawaban. Untuk memberikan dukungan agar efiisien seperti yang diinginkan, dipersiapkan *piggy-backing* (celengan), masing-masing frame data dilengkapi dengan daerah yang menangkap urutan nomer dari frame, ditambah daerah yang menangkap urutan nomer yang dipakai sebagai jawaban. Selanjutnya bila suatu station memiliki data yang akan dikirim dan jawaban yang akan dikirimkan, maka dikirimkan bersama-sama dalam satu frame, cara yang demikian dapat meningkatkan kapasitas komunikasi. Jika suatu *station* memiliki jawaban tetapi tidak memiliki data yang akan dikirim, maka *station* tersebut mengirimkan frame jawaban yang terpisah. Jika suatu *station* memiliki data yang akan dikirimkan tetapi tidak memiliki jawaban baru yang akan dikirim maka station tersebut mengulangi dengan mengirimkan jawaban terakhir yang dikirim, hal ini disebabkan frame data dilengkapi daerah untuk nomer jawaban, dengan suatu nilai (angka) yang harus diletakkan kedalam daerah tersebut. Jika suatu station menerima jawaban yang sama (duplikat) maka tinggal mengabaikan jawaban tersebut. *Sliding-window* dikatakan lebih efisien karena jalur komunikasi disiapkan seperti pipa saluran yang setiap saat dapat diisi beberapa frame yang sedang berjalan, tetapi pada *stop-and-wait* hanya satu frame saja yang boleh mengalir dalam pipa saluran tersebut.

7.3. PROTOKOL SDLC

Protokol *synchronous-data-link-control* (SDLC) pertama kali di publikasikan pada pertengahan tahun 1970 oleh IBM untuk mengatasi transfer data antar komputer dengan jumlah data yang besar. Bentuk dasar dari format SDLC adalah frame yang berisi sekelompok bit (*7-bit flag*) yang dipakai sebagai sinkronisasi, yaitu 0111117 yang diletakkan pada awal dan akhir frame, seperti terlihat pada gambar 7.5 dibawah.



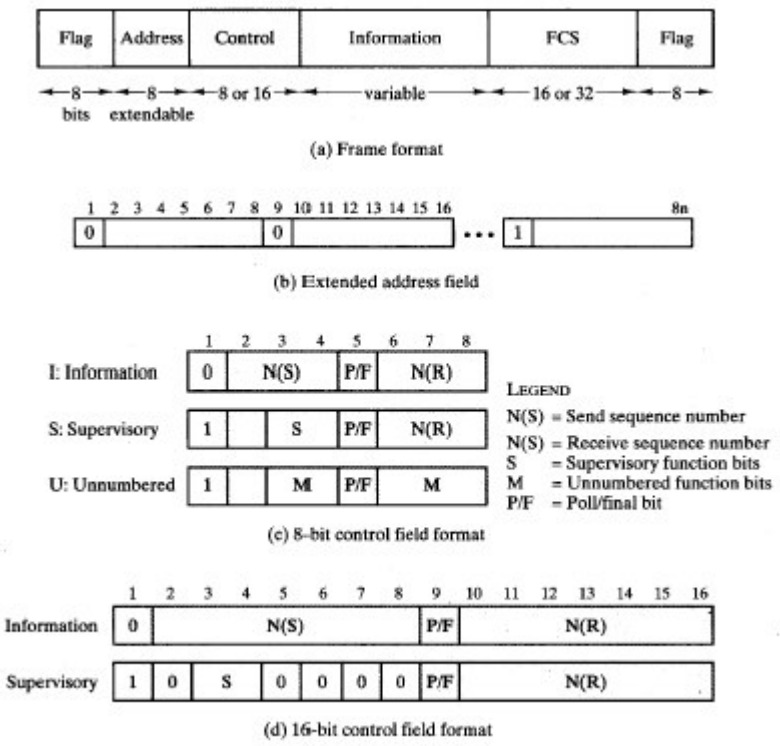
Gambar 7.5 *Format Frame SDLC*

Transmitter menggunakan suatu teknik yang sidebut *bit-stuffing* untuk menghilangkan terjadinya *bit-flag* pada semua frame yang dikirim. Data yang berapa antara *flag* dibaca dan disisipkan bit 0 sesudah terjadi urutan 1 sebaanyak 5 kali, kemudian receiver akan dapat mengetahui awal dan akhir *flag* dengan urutan bit 1 sebanyak 7 kali, dan menghilangkan satu bit 0 sesudah terjadi urutan 1 sebanyak 5 kali, sedangkan bagian frame yang lain disimpan sebagai data.

SDLC dilengkapi dengan sinyal yang ditetapkan sebagai pemberitahu receiver untuk membatalkan frame yang sedang diproses, dengan cata tansmitter mengirimkan 7 urutan bit 1 kepada receiver yang diartikan sebagai karakter pembatal, proses penerimaan data akan terhenti sampai menunggu tanda berikutnya, dan frame data yang sedang diproses dikosongkan.

7.4. PROTOKOL HDLC

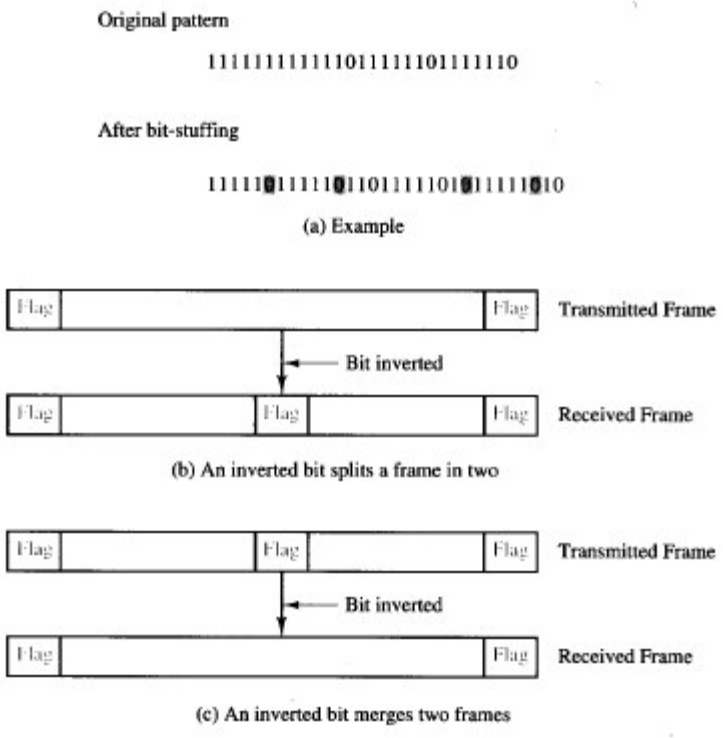
Salah satu protokol untuk *data-link-control* yang paling penting adalah *highlevel- data-link-control* (HDLC) berdasarkan ISO33009 dan ISO4335, yang diadopsi dari standart CCITT untuk *jaringan packet-switching* X.25.



Gambar 7.6 *Struktur Frame HDLC*

Pada awal dan akhir frame pada HDLC juga ditandai dengan menggunakan urutan bit 0111117 seperti pada SDLC, sedangkan perbedaan antara SDLC dan HDLC adalah :

- HDLC menggunakan deretan bit untuk alamat dan controk sebanyak 7 bit
- Karakter pembatal pada HDLC menggunakan 7 bit 1 (SDLC 7 bit)
- Byte terakhir pada field alamat dan pengontrol yang memiliki LSB 1 yang menandakan akhir field.



Gambar 7.7 *Bit-stuffing*

Flag-field
Dipakai untuk pembatas frame pada kedua ujung frame dengan deretan biner 0111117, *flag* tunggal dapat digunakan untuk menutup flag satu frame dan membuka flag untuk frame berikutnya. Interface pada sisi receiver terus menerus mencari urutan *flag* untuk mensinkronkan awal frame. Sambil menerima frame receiver akan terus mencari urutan *bit-flag* untuk menentukan akhir frame.

Address-field
Merupakan identitas station sekunder bagi transmitter dan receiver yang mengirimkan dan menerima frame. *Address-field* biasanya menggunakan 7 bit tetapi disini digunakan 7 bit, seperti pada gambar 7.7b.

Control-field
HDLC menggunakan tiga macam *control-field* yang berbeda.

Information frame (I-frame)
membawa data yang akan dikirimkan, dengan menggunakan mekanisme ARQ sebagai pengatur aliran dan pengontrolan error pada data, sengan piggyback pada information-frame.

Supervisory-frame (S-frame)
menyediakan mekanisme ARQ jika *piggyback* tidak dipakai.

Unnumbered-frame (U-frame)
menyediakan fungsi pengontrol sambungan tambahan. Bit ke satuatau kedua dari *control-field* ini menunjukkan tipe frame, seperti pada gambar 7.7c.
Information-field
Berada pada *I-frame* atau *U-frame*. Field ini berisi deretan 7 bit dengan panjang kelipatan bilangan bulat.

- **Frame-check-sequence (FCS)**
adalah kode pengecekan error yang diperoleh dari perhitungan menggunakan CRC. Pada gambar 7.7 memberikan contoh *bit-stuffing*, pada dua kasus pertama ekstra 0 tidak begitu berpengaruh untuk mengabaikan flag pattern, tetapi penting untuk algoritma operasional

BAB IX

TEKNIK SWITCHING

Komunikasi voice ataupun data tidak terlepas dari teknik switching. Berikut adalah uraian beberapa teknik switching yang diterapkan. Teknik Switching dikenal ada dua buah yaitu *Circuit Switching* and *Packet Switching*.

8.1 Circuit Switching

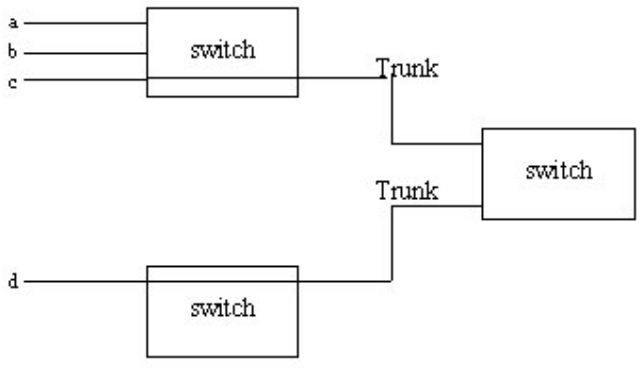
Menerapkan sebuah path komunikasi yang *dedicated* (permanen) antara 2 buah station

- melibatkan tiga fase :
  - *Circuit Establishment*
  - *Signal Transfer* (mungkin analog voice, digitized voice, binary data)
  - *Circuit disconnect*
- kurang efisien karena koneksi tetap *established* walaupun tidak ada data yang ditransfer
- contoh konkret adalah *public telephone network*, PBX (Public Branches eXchange utk gedung)
- tidak complex dalam routing, *flow control*, dan syarat-syarat *error control*

8.2 Routing dalam Circuit Switching

Efisiensi jaringan diperoleh dengan cara meminimisasi switching and kapasitas transmisi. Komponen dalam arsitektur jaringan telekomunikasi umum adalah :

- *pelanggan*
- *local loop* : *link* antara pelanggan dan jaringan. Hampir semuanya menggunakan twisted pair. Panjangnya antara beberapa kilometer dan beberapa puluh kilometer.
- *exchanges* : switching lokal dalam sebuah jaringan.
- Switching Lokal mendukung pelanggan-pelanggan yang dikenal dengan nama *end office* yang biasanya dapat mendukung beribu-ribu pelanggan dalam *local area*.
- *trunks* : cabang-cabang antara *exchanges*. *Trunks* membawa *multiple voice-frequency* dengan menggunakan FDM (*Frequency Division Multiplex*) atau *synchronous TDM (Time Division Multiplex*).



Gambar 8.1 Proses Routing

a dan b koneksi dalam satu buah *end office*, sedangkan c dan d koneksi yang lebih kompleks. Lebih disukai menggunakan *dynamic routing* daripada *static routing*

dikarenakan kondisi traffic yang makin kompleks dan lebih fleksibel. Adapun dalam kelas-kelas dalam *dynamic routing* adalah sebagai berikut :

1. Alternate Routing

Adalah routing-routing pilihan yang dapat digunakan antara dua *end office*. Tiap switch diberikan sejumlah route untuk mencapai tiap tujuan. Jika hanya ada satu route dalam tiap pasang *source-destination*, ini disebut dengan *fixed alternate routing*. Yang lebih umum digunakan adalah *dynamic alternate routing*. *Routing decision* didasari atas *status current traffic* (akan ditolak jika sibuk) dan *historical traffic patterns* (urutan-urutan route yang diinginkan).

2. Adaptive Routing

Didesain untuk memfungsikan switch dalam mengubah bentuk *traffic* pada sebuah jaringan. Situasi seperti ini, switch yang ada saling bertukar informasi untuk mempelajari kondisi jaringan sehingga tipe routing ini lebih efisien daripada *alternate routing* dalam hal *resourcing* jaringan.

DTM (*Dynamic Traffic Management*) yang dikembangkan oleh Northern Telecom menggunakan *central network* untuk mencari *the best alternate route* bergantung dari *congestion* (kepadatan) dalam jaringan tersebut. *Central controller* mengumpulkan status data dari tiap switch untuk mencari alternate route yang diinginkan.

Jaringan dengan menggunakan *circuit-switched* adalah didesain untuk *voice traffic*. Walaupun demikian, *circuit-switched network* juga digunakan dalam komunikasi data dimana akan terjadi :

- untuk *terminal-to-host data connection*, waktu pada line terbuang percuma. Jadi komunikasi data akan tidak efisien jika menggunakan *circuit-switched network*.
- koneksi menyediakan rate yang konstan. Jadi device yang saling terhubung mempunyai rate yang sama saat *transmit* atau *receiving data*. Ini membatasi utilitas dalam jaringan yang banyak terdapat variasi komputer dan terminal.

8. 3. Packet Switching

Dalam *Packet Switching*, data yang ditransmisikan dibagi-bagi ke dalam paket-paket kecil. Jika *source* mempunyai *message* yang lebih panjang untuk dikirim, *message* itu akan dipecah ke dalam barisan-barisan paket. Tiap paket berisi data dari user dan *info control*. *Info control* berisi minimal adalah info agar bagaimana paket bisa melalui jaringan dan mencapai alamat tujuan.

Beberapa keuntungan yang diperoleh dari *packet switching* :

- efisiensi *line* sangat tinggi; hubungan *single node-to-node* dapat di*share* secara dinamis oleh banyak paket. Paket-paket di*queue* dan ditransmisikan secepat mungkin. Secara kontras, dalam *circuit switching*, waktu pada *link node-to-node* adalah dialokasikan terlebih dahulu menggunakan *time-division multiplexing*.
- jaringan packet-switched dapat membuat konversi *data-rate*. Dua buah station yang berbeda *data-rate*nya dapat saling menukar paket.
- ketika traffic mulai padat, beberapa *call* diblok, yang menunjukkan jaringan menolak permintaan koneksi tambahan sampai beban di jaringan menurun. Dalam *packet switchied network*, paket masih dapat diterima akan tetapi *delay delivery* bertambah.
- prioritas dapat digunakan. Jadi kalau sebuah node mempunyai sejumlah *queued packet* untuk ditransmisikan, paket dapat ditransmisikan pertama kali berdasarkan prioritas yang lebih tinggi. Paket-paket ini mempunyai delay yang lebih kecil daripada *lower-priority packets.*

8.4. Operasi Internal

Ada dua pendekatan yang berhubungan dengan jaringan, yaitu *datagram* dan *virtual circuit*. Pada *datagram* tiap paket bisa diroutekan berbeda, misalnya station A akan kirim paket 1, 2, dan 3. Route A menuju E ada dua route, maka kemungkinan paket 1 menempuh route yang berbeda dengan paket 2 tergantung dari kepadatan masing-masing jalur. Sedangkan pada *virtual circuit*, sebuah route antara station dikonfigurasi sebelum terjadi transfer data. Ini bukan *dedicated path* seperti dalam *circuit-switching*. Sebuah paket masih disimpan dalam tiap node. Perbedaannya dengan *datagram* adalah node tidak perlu melakukan *routing decision* untuk tiap paket, dilakukan hanya sekali dan berlaku untuk semua paket.

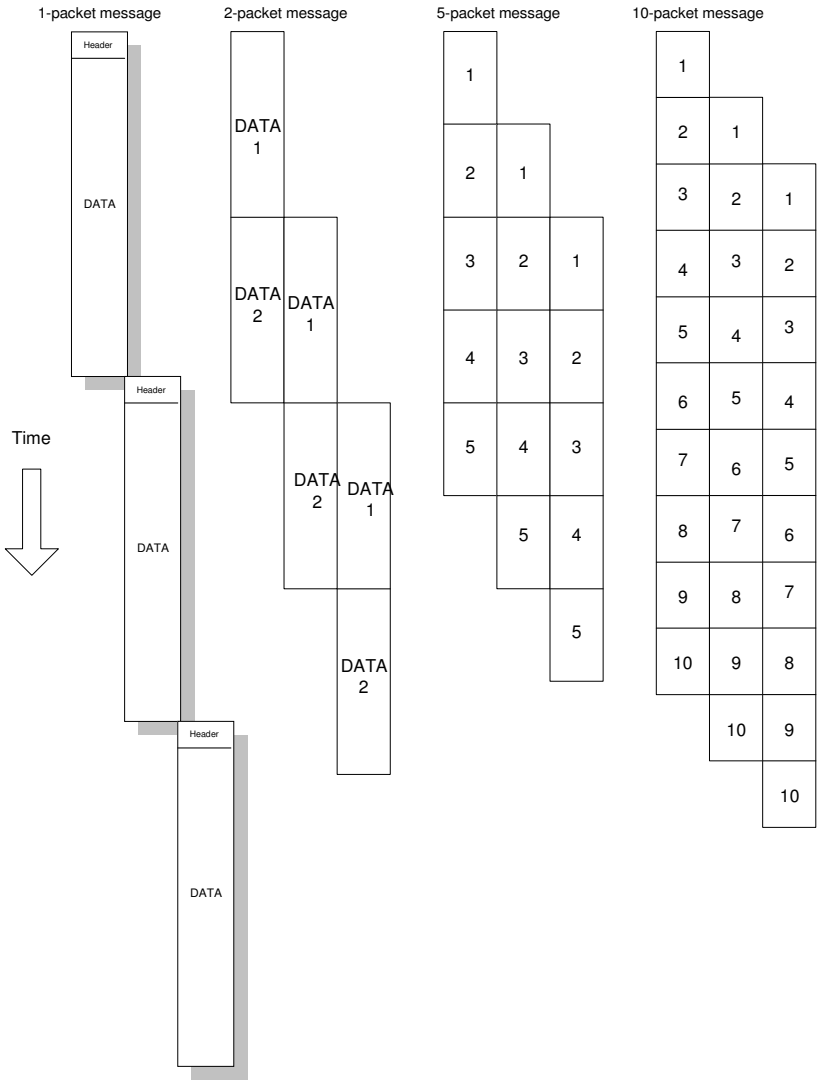
Jika ada dua station yang akan saling menukar data dalam periode waktu tertentu, maka dapat dipastikan keuntungan banyak diperoleh jika menggunakan *virtual circuit*. Pertama, jaringan menyediakan pelayanan yang berhubungan dengan *virtual circuit* termasuk *sequencing* and *error-control*. *Sequencing* berfungsi apabila semua paket mengambil route yang sama. *Error control* adalah pelayanan untuk meyakinkan semua paket dapat tiba di tujuan, tapi juga tiba dengan paket yang benar-benar diinginkan, tidak ada cacat.

Keuntungan dari datagram adalah *call setup phrase* dapat dihindari. Jadi sebuah station yang mengirim hanya satu atau sedikit paket pengiriman *datagram* akan lebih cepat. Keuntungan yang lain adalah lebih *flexible*, lebih *primitive*. Sebagai contoh, apabila ada satu bagian network yang buntu, maka *datagram* yang dikirim akan mengambil route menjauhi network tersebut. Dengan penggunaan *virtual circuit*, karena paket-paket didefinisikan *routing*nya sebelum dikirim maka hal ini akan menjadi sulit apabila route yang diambil mengalami buntu. Keuntungan ketiga adalah pengiriman *datagram* secara tersirat lebih *reliable*. Pada *virtual circuit*, apabila ada node yang gagal, semua *virtual circuit* yang mendefinisikan lewat node tersebut akan lenyap. Pada *datagram*, paket-paket akan mencari alternatif routing dimana akan mengabaikan node yang gagal. Di *virtual circuit* pada operasi internalnya digunakan *packet-switching*.

Dari sudut pandang user, tidak akan dapat begitu berbeda apabila provider menggunakan *packet-switched* atau *circuit-switched network*.

8.5. Ukuran Paket

Ada hubungan antara ukuran paket dengan waktu dalam pentransmisian data. Pada gambar terlihat bahwa data apabila dipecah makin kecil membutuhkan waktu lebih cepat, dan tiap paket pecahannya harus disisipi *heade*mya. Akan tetapi jika dipecah semakin kecil akan didapatkan waktu transmisi yang lebih besar dari sebelum paket lebih diperkecil lagi. Dalam hal ini harus dipilih pemecahan paket yang optimum.



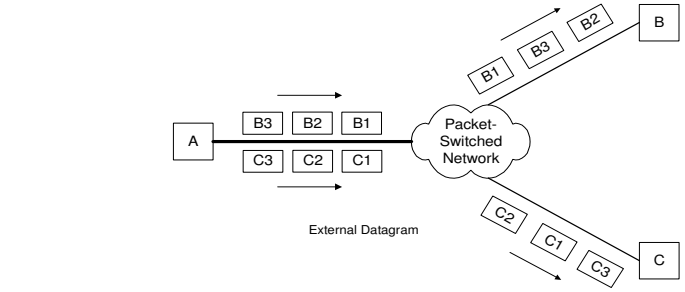
Gambar 8.2 Perbedaan Ukuran-Ukuran Paket

8.6. Operasi Internal dan External Service

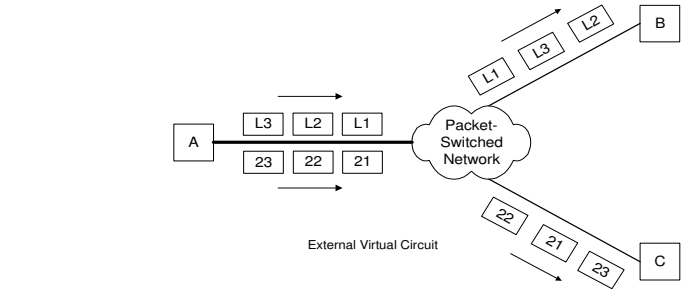
Hal terpenting dalam *packet-switched network* adalah pemilihan dalam menggunakan *datagram* atau *virtual circuit*. Pada interface antara sebuah station dengan sebuah node network, network harus menyediakan pelayanan *connection-oriented* dan *connection-less*. Pada *connection-oriented*, sebuah station melakukan *call request* untuk membentuk sebuah *logical connection* ke station yang lain. Semua paket yang disajikan ke dalam network diidentifikasi kepunyaan *logical connection* tertentu dan diberi nomor secara berurut.

Logical connection biasanya merujuk pada sebuah pelayanan *external virtual circuit* yang jauh berbeda dari konsep operasi *internal virtual circuit*. Sedangkan pada pelayanan *connectionless*, jaringan hanya menangani paket secara *independent* dan mungkin tidak ditransmisikan secara berurut. Tipe service seperti ini dikenal dengan nama *external datagram service* yang juga jauh berbeda dari konsep operasi *internal datagram service*. Secara internal, jaringan akan membuat route antara *endpoints* (*virtual circuit*) atau tidak (datagram).

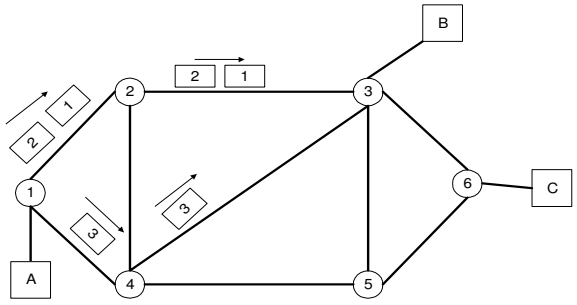
- *External virtual circuit, internal virtual circuit* : Jika user meminta *virtual circuit*, sebuah *dedicated route* yang melintasi dalam jaringan akan dibangun. Semua paket mengikuti route yang sama.
- *External virtual circuit, internal datagram* : Jaringan menangani tiap paket secara terpisah. Jadi, paket-paket yang berbeda dalam *external virtual circuit* yang sama akan mengambil route yang mungkin berbeda.
- *External datagram, internal datagram* : Tiap paket diperlakukan secara bebas dari segi user atau dari segi jaringannya.
- *External datagram, internal virtual circuit*



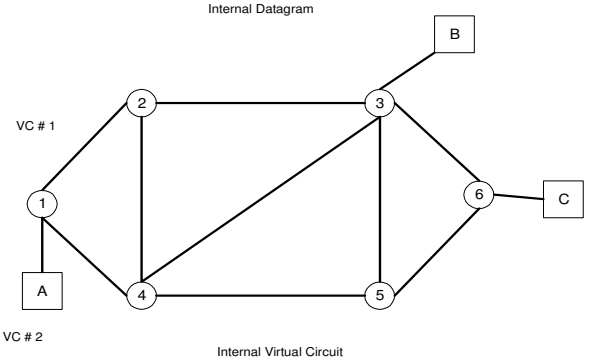
External Datagram



External Virtual Circuit



Internal Datagram



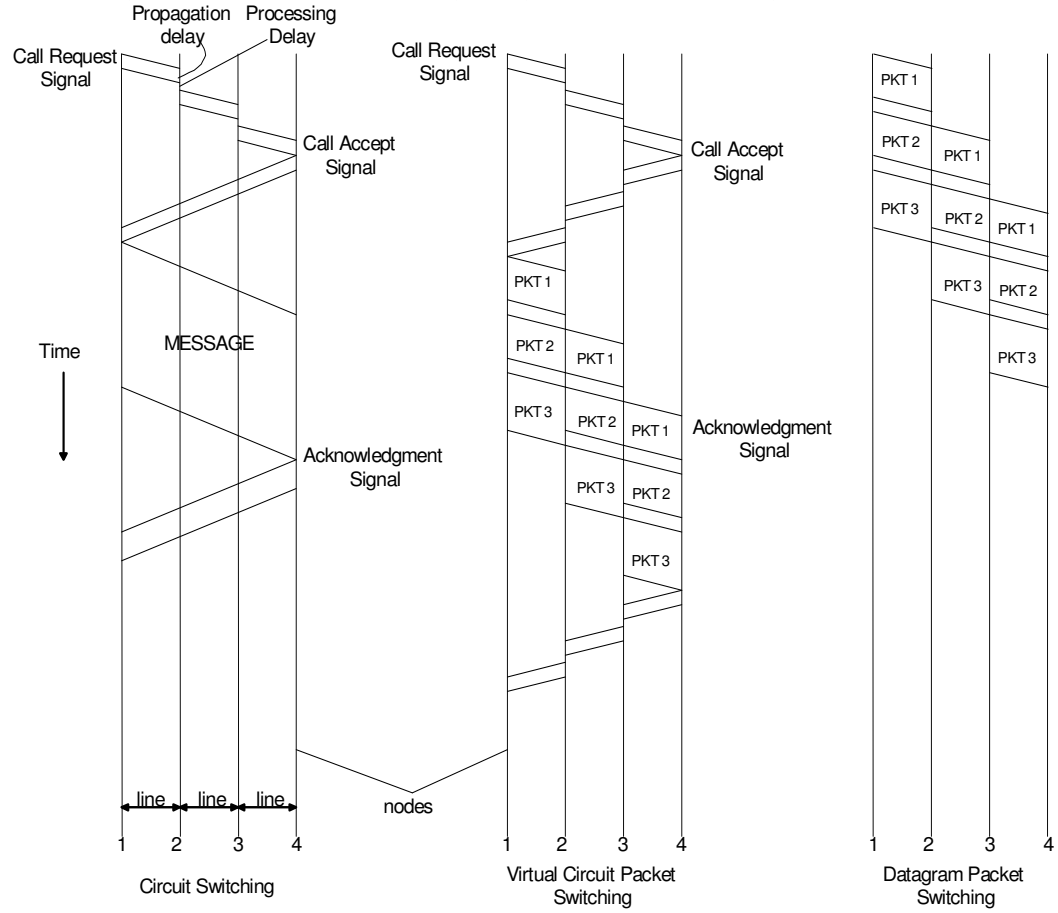
Internal Virtual Circuit

Gambar 8.3 Perbedaan antara External dan Internal Operation

Pemilihan akan *virtual circuit* dengan *datagram* tergantung dari desain objek untuk komunikasi jaringan dan faktor-faktor *cost* secara detailnya.
Untuk *external service* :

- *datagram service* memberikan penggunaan yang efisien dari jaringan dimana tidak ada *call setup*. Ini akan cocok untuk penggunaan beberapa aplikasi real time.
- *virtual circuit service* dapat menyediakan *end-to-end sequencing* dan *error control*. Ini akan cocok untuk aplikasi seperti *file transfer* dan *remote access terminal*.

Perbandingan antara *Circuit Switching* dengan *Packet Switching* :



Gambar 8.1 Perbedaan antara Circuit Switching dengan Packet Switching

Pada gambar 8.1 dimisalkan ada 4 node, node 1 sebagai *source address* dan node 4 sebagai *destination address*. Untuk *circuit switching* ada sejumlah delay sebelum *message* dikirim, yaitu untuk *call request*, lalu jika *destination station* tidak sibuk, sinyal *accepted* dikirim dari *destination address*. Proses ini tidak berlangsung setelah koneksi telah disetup. *Virtual circuit switching* hampir sama dengan *circuit switching*. Berbeda dengan *circuit switching*, *call acceptance* akan memakan waktu (delay) walaupun koneksi telah *established*. Hal itu karena paket itu mengalami antrian dan harus menunggu untuk retransmisi. Sekali *virtual circuit* established, *message* akan dikirim dalam bentuk paket-paket. Maka *virtual circuit* tidak akan lebih cepat dari *circuit switching*.

Datagram packet switching tidak membutuhkan *call setup*. Jadi untuk *message* pendek akan lebih cepat dari *virtual circuit packet switching* dan mungkin juga *circuit switching*. Selama tiap *datagram* di*route* secara bebas, proses untuk tiap *datagram* di tiap node mungkin lebih panjang dari paket-paket *virtual circuit*. Jadi untuk message yang panjang-panjang, teknik *virtual circuit* mungkin diutamakan.

Tabel 8.1. Perbedaan antara Circuit Switching dengan Packet Switching

Circuit Switching	Virtual-Circuit Packet Switching	Datagram Packet Switching
Dedicated transmission path	No dedicated path	No dedicated path
Continous transmission of data	Transmission of packets	Transmission of packets
Fast enough for interactive	Fast enough for interactive	Fast enough for interactive
Messages are not stored	Packets stored until delivered	Packets may be stored until delivered
The path is established for entire conversation	Route established for entire conversation	Route established for each packet
Call setup delay; negligible transmission delay	Call setup delay; packet transmission delay	Packet transmission delay
Busy signal if called party busy	Sender notified of connection denial	Sender may be notified if packet not delivered
Overload may block call setup; no delay for established calls	Overload may block call setup; increases packet delay	Overload increases packet delay
User responsible for message loss protection	Network may be responsible for packet sequences	Network may be responsible for individual packets
Usually no speed or code conversion	Speed and code conversion	Speed and code conversion
Fixed-bandwidth transmission	Dynamic use of bandwidth	Dynamic use of bandwidth
No overhead bits after call setup	Overhead bits in each packet	Overhead bits in each packet