BAB II

LANDASAN TEORI

Landasan teori adalah teori-teori yang *relevan* dan dapat digunakan untuk menjelaskan variabel-variabel penelitian. Landasan teori ini juga berfungsi sebagai dasar untuk memberi jawaban sementara terhadap rumusan yang diajukan serta membantu dalam penyusunan *instrumen*t penelitian.

2.1 Kompresi Data

Kompresi berarti memampatkan/mengecilkan ukuran. Kompresi data adalah proses mengkodekan informasi menggunakan bit atau *information-bearing unit* yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem *encoding* tertentu. Contoh kompresi sederhana yang biasa kita lakukan misalnya adalah menyingkat kata-kata yang sering digunakan tapi sudah memiliki konvensi umum. Misalnya: kata “yang” dikompres menjadi kata “yg”.

Kompresi data teks dan citra *digital* adalah proses untuk meminimalisasi jumlah bit yang merepresentasikan suatu data baik berupa teks maupun citra/gambar sehingga ukuran data menjadi lebih kecil. Pengiriman data hasil kompresi dapat dilakukan jika pihak pengirim atau pihak penerima memiliki aturan yang sama dalam hal kompresi data. Pihak pengirim harus menggunakan algoritma kompresi data yang sudah baku dan pihak penerima juga menggunakan teknik dekompresi data yang sama dengan pengirim sehingga data yang diterima

dapat dibaca dan di-*decode* kembali dengan benar. Kompresi data menjadi sangat penting karena memperkecil kebutuhan penyimpanan data, mempercepat pengiriman data, memperkecil kebutuhan *bandwidth*.

Dalam teknik kompresi data, besarnya kapasitas data pada saat pengiriman, dan media penyimpanan yang terbatas menjadi masalah utama. Kompresi data ditujukan untuk mereduksi penyimpanan data yang redundan atau dalam istilah lain kompresi data teks dan citra *digital* dilakukan untuk dengan cara meminimalkan jumlah bit yang diperlukan untuk merepresentasikan suatu data, namun seringkali kualitas data teks dan citra yang dihasilkan jauh lebih buruk dari aslinya karena keinginan kita untuk memperoleh rasio kompresi yang tinggi.

Secara mendasar standar-standar teknik kompresi data memiliki 2 teknik kompresi yaitu teknik kompresi secara *lossy* dan *lossless.* Untuk teknik kompresi secara *lossy* ada sebagian dari informasi yang hilang dalam data tersebut, sehingga jika dilakukan proses *edit* akan susah dilakukan. Sedangkan teknik kompresi secara *lossless* tidak menghilang informasi sedikitpun.

2.2 Faktor Penting Kompresi Data

Dalam kompresi data, terdapat 4 (empat) faktor penting yang perlu diperhatikan, yaitu :

a. *Time Process* (waktu yang dibutuhkan dalam menjalankan proses),
b. *Completeness* (kelengkapan data setelah *File-File* tersebut dikompres),
c. *Ratio compress* (ukuran data setelah dilakukan kompresi),

d. *Optimaly* (perbandingan apakah ukuran *File* sebelum dikompres sama atau tidak sama dengan *File* yang telah dikompres). Tidak ada metode kompresi yang paling *efektif* untuk semua jenis *File.*

2.2.1 Metode Kompresi Data

Berdasarkan tipe peta kode kompres data yang digunakan untuk mengubah pesan awal (isi *File input*) menjadi sekumpulan *code word*, metode kompresi terbagi menjadi dua kelompok, yaitu:

a. Metode statik : menggunakan peta kode yang selalu sama. Metode ini membutuhkan dua *fase* (*two-pass*): *fase* pertama untuk menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya dan *fase* kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan. Contoh : algoritma *Huffman Static.*

b. Metode dinamik (*adaptif*) : menggunakan peta kode yang dapat berubah dari waktu ke waktu. Metode ini disebut adaptif karena peta kode mampu beradaptasi terhadap karakteristik isi *File* selama proses kompresi berlangsung. Metode ini bersifat *onepass*, karena isi *File* selama dikompres hanya diperlukan satu kali pembacaan terhadap isi *File.* Contoh algoritma *Lemple Ziv Welch* (LZW) dan *Dinamic Markov Compression* (DMC).

Berdasarkan teknik pengkodean/ pengubahan simbol yang digunakan, metode kompresi dapat dibagi ke dalam tiga kategori, yaitu :

a. Metode *simbolwise* : metode yang menghitung peluang kemunculan karakter dalam data teks, kemudian karakter tersebut dikodekan dimana karakter yang sering muncul diberi kode lebih pendek dibandingkan karakter yang jarang muncul contoh : algoritma *Huffman*.

b. Metode *Dictionary* : menggantikan karakter/fragmen dalam *File input* dangan indeks lokasi dari karakter/*fragmen* tersebut dalam sebuah kamus (*dictionary*), contoh : algoritma LZSS.

c. Metode *Predictive* : menggunakan model *finite-context* atau *finite-state* untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya, contoh : algoritma DMC.

2.3 Sifat Kompresi berdasarkan hasil

Algoritma kompresi diklasifikasikan menjadi dua buah yaitu:

2.3.1 Lossy Compression

Keuntungan dari algoritma ini adalah bahwa rasio kompresi (perbandingan antara ukuran data yang telah dikompresi dengan data sebelum dikompresi) cukup tinggi. Namun algoritma ini dapat menyebabkan data pada suatu berkas yang dikompresi hilang ketika didekompresi. Hal ini dikarenakan cara kerja *lossy* adalah dengan mengeliminasikan beberapa data dari suatu *File.* Namun data yang dieliminasikan biasanya adalah data yang kurang diperhatikan atau di luar jangkauan manusia, sehingga pengeliminasian data tersebut

kemungkinan besar tidak akan mempengaruhi manusia yang berinteraksi dengan *File* tersebut. Contohnya pada pengkompresian *File* teks, kompresi *lossy* akan mengeliminasi data dari *File* teks yang memiliki frekuensi sangat tinggi/rendah sehingga banyak menghilangkan informasi dalam data yang dianggap penting. Beberapa jenis data yang biasanya masih dapat mentoleransi algoritma *lossy* adalah citra, audio dan video.

2.3.2 Lossless Compression

Berbeda dengan algoritma kompresi *lossy*, pada algoritma kompresi *lossless*, tidak terdapat perubahan data ketika mendekompresi *File* yang telah dikompresi dengan kompresi *lossless* ini. Algoritma ini biasanya diimplementasikan pada kompresi *File text*, seperti program komputer (*File* zip, rar, gzip, dan lain-lain).

2.4 File Teks

File teks adalah arsip yang disimpan dalam suatu media *corage* berupa data yang terdiri dari karakter yang menyatukan kata-kata / *symbol*. File *text* terpakai sebagai penyimpanan yang memiliki sifat dan tidak memiliki organisasi data yang jelas melakukan proses yang kompleks untuk melakukan pengelompokan data.

2.4.1 Konsep Dasar Teknik Kompresi Data Teks

Teknik kompresi data berupa teks adalah kumpulan dari karakter-karakter atau *string* yang menjadi satu kesatuan. Teks yang memuat banyak karakter didalamnya selalu menimbulkan masalah pada edia penyimpanan dan kecepatan waktu pada saat transmisi data. Media penyimpanan yang terbatas dan waktu yang lama dalam melakukan *transfer* data teks, membuat orang mencoba berpikir untuk menemukan cara yang dapat digunakan untuk mengkompres data teks.

Ide dasar dari kompresi data teks adalah mengkodekan kembali setiap karakter didalam dokumen atau pesan dengan kode yang lebih pendek sehingga dapat memperkecil ukuran dokumen. Dokumen atau *File* teks yang dibuat di dalam komputer atau media elektronik lain seperti *.doc,docx .pdf, .rtf* merupakan kumpulan kode yang mewakili setiap karakter dalam dokumen atau pesan. Kode yang biasa dan banyak digunakan adalah kode ASCII (*American Standard Code for Information Interchange*).

a. Misalnya pada teknik kompresi *Run Length Encoding* (RLE)

Contoh :

a. Data: ABCCCCCCCCDEFGGGG = 17 karakter
b. RLE tipe 1 (min. 4 huruf sama) : ABC!8DEFG!4 = 11 karakter
c. RLE ada yang menggunakan suatu karakter yang tidak digunakan dalam teks tersebut seperti misalnya ‘!’ untuk menandai.

d. RLE ada yang menggunakan *flag* bilangan *negatif* untuk menandai batas sebanyak jumlah karakter tersebut.

e. Berguna untuk data yang banyak memiliki kesamaan, misal teks ataupun grafik seperti *icon* atau gambar garis-garis yang banyak memiliki kesamaan pola.

f. *Best case*: untuk RLE tipe 2 adalah ketika terdapat 127 karakter yang sama sehingga akan dikompres menjadi 2 *byte* saja.

g. *Worst case*: untuk RLE tipe 2 adalah ketika terdapat 127 karakter yang berbeda semua, maka akan terdapat 1 *byte* tambahan sebagai tanda jumlah karakter yang tidak sama tersebut.

b. Teknik Kompresi Data

Teknik kompresi data terdapat 4 cara dalam melakukan kompresi yaitu sebagai berikut :

a. Kompresi berbasis Statistik (*Lossless*) yaitu Merepresentasikan citra dengan *frekuensi* kemunculan nilai intensitas tertentu.

b. Kompresi berbasis *Kuantisasi* (*Lossy*) yaitu Mengurangi jumlah intensitas warna.

c. Kompresi berbasis *Transformasi* (*Lossless/Lossy*) yaitu Mengoptimalkan kinerja kompresi berbasis statistik dan kuantisasi dengan cara melakukan transformasi terlebih dahulu sebelum menerapkan salah satu teknik tersebut. Sehingga kompresi bersifat *lossy* atau *lossles* tergantung teknik mana yang digunakan setelah *transformasi* apakah itu statistik (*lossless*) atau kuantisasi *(lossy*).

d. Kompresi berbasis *Fraktal* (*Lossy*) yaitu merupakan bentuk *rekursif* yang merepresentasikan komponen dasar objek. Dalam konsep kompresi, data direpresentasikan sebagai pasangan antar elemen *fraktal*, pola umum konfigurasi yang membentuk objek secara keseluruhan, dan koefisien transformasi spasial (*affine*) untuk masing-masing *fraktal* sesuai dengan posisinya dalam konfigurasi pembentuk objek.

2.5 Konsep Dasar Teknik kompresi citra *digital*

Teknik kompresi citra mengacu pada dua konsep dasar, yaitu :

1. Mengeksploitasi redundansi informasi yang terdapat pada pola sinyal citra *digital*. Metode ini digunakan pada teknik kompresi citra *lossless coding*.

Redundansi tersebut dapat berupa:

a. Redundansi *Spasial* akibat korelasi antara piksel-piksel yang bertetangga yang memiliki intensitas yang sama
b. Redundansi *Spektral* akibat korelasi antara bidang-bidang warna yang berbeda
c. Redundansi *Temporal* akibat korelasi *frame-frame* yang berbeda pada citra dinamis.
d. Redundansi *Statistik* akibat *redundansi ruang dan redundansi spectrum.*

e. Redundansi *Psikovisual* akibat danya kenyataan bahwa mata manusia tidak sensitif terhadap adanya *frekuensi* beberapa ruang dalam citra.

2. Menggunakan deviasi dalam batas yang dapat ditoleransi dengan cara mengurangi detail citra yang tidak dapat ditangkap oleh penglihatan manusia. *Resolusi* spasial, waktu dan amplitudo disesuaikan dengan aplikasi yang digunakan. Metode ini digunakan pada teknik kompresi citra *lossy coding* dengan *mengeksploitasi redundansi* statistik dan *psikovisual* .

2.5.1 Kriteria penilaian kualitas citra *digital*

Pada bagian ini dibahas mengenai criteria-kriteria penilaian kualitas baik buruknya suatu citra dan kriteria penilaian dibagi menjadi dua yaitu kriteria penilaian objektif dan kriteria penilaian *subjektif*.

1. Kriteria penilaian *objektif*

Kriteria penilaian *Objektif* ini didasarkan pada batas *error* yang diperbolehkan untuk citra yang akan diolah. Untuk citra *f(x,y)* dan citra hasil proses *g(x,y),* maka beberapa parameter yang dijadikan sebagai kriteria penilaian objektif adalah sebagai berikut :

a) *Peak Signal to Noise Ratio* (PSNR), kita dapat membuat ukuran kualitas hasil pemampatan citra menjadi ukuran *kuantitatif* dengan menggunakan PSNR. PSNR dihitung untuk mengukur perbedaan antara citra semula dengan citra hasil pemampatan dengan citra semula, dengan rumus:

$$PSNR = 10\log(255/MSE)^2 \qquad (2.1)$$

b) *Mean Square Error* (MSE) :

$$MSE = \frac{1}{MN} \sum_{X=0}^{M=X} \sum_{Y=0}^{N=Y} (citra_hasil(x,y) - citra_asli(x,y))^2 \text{ ..... (2.2)}$$

c) *Mean Absoluted Error* (MAE) :

$$MAE = \frac{1}{MN} \sum_{X=0}^{M=X} \sum_{Y=0}^{N=Y} \left| citra_hasil(x,y) - citra_asli(x,y) \right| \text{ ..... (2.3)}$$

d) *Number of Region* (NR) :

Yaitu jumlah *region* atau segmen hasil segmentasi.

e) *Time* (t) :

Yaitu lama proses yang dibutuhkan dalam proses segmentasi.

2. Kriteria penilaian *subjektif*

Kriteria ini ditentukan berdasarkan hasil pengamatan oleh mata manusia. Dengan menggunakan kriteria ini, baik buruknya citra hasil pengolahan ditentukan oleh pengamat sendiri. Sebagai contoh, dua buah citra yang memiliki salah satu yang sama pada kriteria penilaian kualitas citra secara *objektif* dapat mempunyai kualitas subjektif yang berbeda, tergantung dari persepsi *visual* pengamat. Beberapa kriteria hasil penilaian subjektif yang banyak digunakan adalah sebagai berikut:

- *Excellent* (skor 9 atau 10)

  Citra hasil segmentasi yang diamati mempunyai kualitas sangat baik, menggambarkan *region* citra dengan tepat atau mendekati tepat.

- *Fine* (skor 7 atau 8)

Citra hasil segmentasi yang diamati masih mempunyai kualitas tinggi, menggambarkan *region* citra dengan sedikit gangguan-gangguan atau kesalahan.

- *Passable* (skor 5 atau 6)

  Citra hasil segmentasi yang diamati mempunyai kualitas agak baik, menggambarkan *region* citra dengan gangguan-gangguan atau kesalahan atau kesalahan yang sedikit berarti.

- *Marginal* (skor 3 atau 4)

  Citra hasil segmentasi yang diamati mempunyai kualitas buruk, menggambarkan *region* citra dengan gangguan yang cukup besar.

- *Inferior* (skor 1 dan 2)

  Citra hasil segmentasi yang diamati mempunyai kualitas yang sangat buruk, tetapi *region* citra masih dapat diamati secara kasar dengan gangguan-gangguan yang sama jelas atau sangat besar.

- *Unusable* (skor 0)

  Citra hasil segmentasi yang diamati sangat buruk sudah tidak dapat diamati lagi.

2.6 Lemple Ziv Storer Symanski (LZSS)

Algoritma *Lemple Ziv Storer Symanski* (LZSS) adalah algoritma kompresi data lossless yang dimodifikasi dari LZ77, dinamai penciptanya *James Storer* dan *Thomas Szymanski* (yang dibangun di atas karya *Abraham Lempel* dan *Jacob* Ziv). Algoritma *Lemple Ziv Storer Symanski* (LZSS) salah satu kompresi urutan simbol data (misalnya, *byte* data) dengan mengidentifikasi urutan simbol

yang berulang dalam masukan, dan menggantikan urutan-urutan simbol yang lebih kecil. Implementasi *Lemple Ziv Storer Symanski* (LZSS) dapat menyesuaikan jumlah bit yang dialokasikan dengan mengganti panjang ukuran *byte* (antara parameter lain) untuk mendapatkan kinerja kompresi yang cukup baik.

2.6.1 Algoritma Lemple Ziv Storer Symanski (LZSS)

Algoritma untuk *Lemple Ziv Storer Symanski* (LZSS) adalah sebagai berikut :

1. Tempatkan posisi koding pada permulaan *input stream.*
2. Cari *longest match* pada *window* untuk *lookahead buffer*;
    a. P = pointer untuk mencocokan;
    b. L = panjang input yang cocok;
3. Apakah L > = MIN_LENGTH?
    a. Jika YA : outputnya P dan bergerak ke posisi L karakter;
    b. Jika TIDAK : outputnya karakter pertama pada *lookhead buffer* dan bergerak satu posisi karakter ke depan;
4. Jika karakter input *stream* banyak, ulangi langkah 2.

2.6.2 Window dan Match Length

Ada beberapa aturan agar sistem dapat bekerja yaitu:

a. Untuk *flag* mengidentifikasi sebuah *literal* atau *match.*
b. Seberapa jauh data sebelumnya dapat dicocokan.
c. Jumlah *bytes* maksimum yang dapat dicocokan.

Untuk aturan yang pertama hanya membutuhkan dua *flag* yaitu *literal* atau *match,* jadi hanya menggunakan bit tunggal. Jika menggunakan bit tunggal maka seluruh *bytes* tidak selalu ditulis tapi hanya sebagian saja. Sebelum proses pemampatan dilakukan, dibutuhkan suatu fungsi kode untuk membaca dan menulis jumlah variable bit dari atau ke data *stream.*

Sebagai contoh : batas *offset* menggunakan 4 bits, jadi range-nya antara 0-31. Sedangkan untuk len menggunakan 3 bits jadi range-nya antara 0-7. Outputnya akan ditulis menjadi (4,3) dalam *byte* menjadi "1 0100 011". 1 bit pertama menunjukan *flag,* diikuti 4 bits berikutnya yang menunjukan *offset,* kemudian 3 bits berikutnya menunjukan *len*.

Contoh penggunaannya :

Input "1231231" dalam biner ASCII ditulis menjadi 56 bits:

00000001 00000010 00000011 00000001

00000010 00000011 00000001

Dikodekan menjadi :

1. *Input* Data = "1231231"

   Mulai dengan *byte* "1". Apakah pernah ada sebelumnya?

   "TIDAK", maka akan dienkodekan sebagai *literal*. *Output* Data = 0 0000001

2. *Input* Data = "1231231"

Byte berikutnya “2”. Apakah pernah ada sebelumnya? “TIDAK”, maka dienkodekan sebagai *literal*.

Output Data = 0 0000001 0 00000010

3. *Input* Data = “1231231”

   Byte berikutnya “3”. Apakah pernah ada sebelumnya? “TIDAK”, maka dienkodekan sebagai *literal*.

   Output Data = 0 0000001 0 00000010 0 00000011

4. *Input* Data = “1231231”

   Byte berikutnya “1”. Apakah pernah ada sebelumnya? “YA”, 3 bytes sebelumnya. Berapa *bytes* yang sama ? 3 bytes “123”, maka outputnya untuk *flag* “1” (1 *bit*) diikuti *offset* “3” (4 *bits*), dan untuk len “3” (3 *bits*). Output Data = 0 0000001 0 00000010 0 00000011 1 0011 011.

5. *Input* Data = “1231231”

   Byte berikutnya “1”. Apakah pernah ada sebelumnya? “YA”, 3 *bytes* sebelumnya. Berapa *bytes* yang sama ? 1 *bytes* “1”, maka outputnya untuk *flag* “1” (1 *bit*) diikuti *offset* “3” (4 *bits*), dan untuk len “1” (3 *bits*). Output Data = 0 0000001 0 00000010 0 00000011 1 0011 011 1 0011 001. *Input* data yang tadinya 56 bits, *output* datanya menjadi 43 bits.

2.6.3 Cyclic Redudancy Checking (CRC)

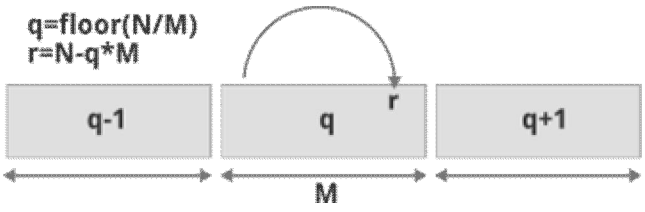
Cyclic Redudancy Checking (CRC) merupakan kaidah untuk meralat data yang akan dikirimkan melalui saluran komunikasi, Pengirim

mengaplikasikan *polynomial* 16 atau 32 bit pada blok data yang akan dikirim dan menambahkan hasil polynomial yang sama pada data dan mebandingkan hasilnya dengan yang diterimanya. Jika keduanya sama maka data telah berhasil dikirim. Jika tidak, pengirim harus mengirimkan kembali data semula yang diralat.

Consultative Committee of the Internasional telephone and Telegraph (CCITI) mempunyai kemampuan polynomial 16 bit yang digunakan protocol CRC. IBM SDLC menggunakan *protocol* CRC-16, CRC akan menghasilkan ralat 1 atau 2 bit dan keberhasilannya adalah 99,998% (HooiYu-Mun, 2000)

2.7 Rice Coding

Algoritma *Rice Coding* adalah algoritma yang diciptakan oleh *Robert F. Rice*, yang menunjukan penggunaan sebuah *subse*t dari turunan *Golomb Coding* untuk menghasilkan sebuah kode sederhana yang mungkinkan *suboptimal* dari kode awalan itu sendiri. Algoritma *Rice Coding* ini digunakan dalam skema pengkodean adaptif, yang akan ditunjukan pada gambar 2.1 yang dapat mengacu hanya menggunakan sebuah *subset* dari turunan *Golomb Coding*.



Gambar 2.1 Pengkodean *Integer* pada *Rice Coding*

Pada gambar diatas *Rice coding* dapat dianggap sebagai kode yang menunjukkan pada posisi *(q),* dan *offset* dalam *(r).* Sehingga posisi *q,* dan *r offset* untuk pengkodean integer *N* pada *Rice Coding* yang menggunakan parameter *M*.

Secara *formal*, dua bagian yang diberikan oleh ekspresi berikut, di mana x adalah nomor yang disandikan :

$$q = \left\lfloor \frac{(x-1)}{M} \right\rfloor$$ Dan $r = x - q\,M - 1$ Hasil akhir terlihat seperti : $(q+1)\,r$ (2.4)

Perhatikan bahwa *r* dapat dari berbagai jumlah bit, dan secara khusus bit *b* hanya untuk *Rice Coding,* dan menghubungkan antara b-1 dan b bit untuk kode *Golomb* (M yaitu bukan kelipatan dari 2):

$$b = \lceil \log_2(M) \rceil$$

(2.5)

Jika $0 \le r < 2^b - M$ gunakan -1 *b* bit untuk mengkodekan *r.*

(2.6)

Jika , $2^b - M \le r < M$ gunakan bit untuk mengkodekan *r b.*

(2.7)

, $b = \log_2(M)$ Jika M adalah hasil kedua dan dapat dikodekan semua nilai r dengan bit b.

2.7.1 Algoritma Rice Coding

Algoritma untuk *Rice Coding* adalah pengkodean biner secara sederhana. Dari penjelasan diatas jika parameter M adalah sebagai berikut :

1. Tetapkan nilai M sebagai *integer*.
2. Lakukan perulangan dari N, sampai jumlah yang akan di *encode* cari:
    1. q = *integer* [N/M]
    2. r = N mod M
3. Tampilkan (*generate*) *codeword*
    1. Format kode ; (q) (r), dimana
    2. Code q
        a. Tulis sebanyak panjang *string* q 1 bit
        b. Tulis sebuah 0 bit
    3. *Code* r
        1. Jika M adalah kelipatan dari 2, *code* r adalah format biner. Jadi diperlukan sebanyak $log_2(M)$ bit.
        2. Jika M adalah bukan kelipatan dari 2, $b = \lceil \log_2(M) \rceil$
            a. Jika $r < 2^b - M$ *code* r adalah biner akan digunakan sebanyak b-1 bit.

b. Jika $r \geq 2^b - M$ jumlah codenya $r + 2^b - M$ ditulis dalam biner yang direpresentasikan sebanyak b bit.

2.8 Jaringan Komputer

Jaringan komputer adalah sebuah kumpulan komputer dan peralatan lainnya yang terhubung. Informasi dan data bergerak melalui kabel-kabel sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data, mencetak pada printer yang sama dan bersama-sama menggunakan *hardware/software* yang terhubung dengan jaringan. Tiap komputer, printer atau *periferal* yang terhubung dengan jaringan disebut *node.* Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan *node.* Sebuah jaringan biasanya terdiri dari dua atau lebih komputer yang saling berhubungan diantara satu dengan yang lain, dan saling berbagi sumber daya misalnya CDROM, Printer, pertukaran *File*, atau memungkinkan untuk saling berkomunikasi secara elektronik. Komputer yang terhubung tersebut, dimungkinkan berhubungan dengan media kabel, saluran telepon, gelombang radio, satelit, atau sinar infra merah.

2.8.1 Jenis - Jenis Jaringan Komputer

a. *Local Area Network* (LAN) / Jaringan Area Lokal.

Sebuah LAN adalah jaringan yang dibatasi oleh area yang relatif kecil, umumnya dibatasi oleh area lingkungan seperti sebuah perkantoran di sebuah gedung, atau sebuah sekolah, dan biasanya tidak jauh dari sekitar 1 km persegi. Beberapa model konfigurasi LAN, satu komputer biasanya menjadi sebuah *File server,* digunakan untuk menyimpan perangkat lunak (*software*) yang mengatur aktifitas jaringan, ataupun sebagai perangkat lunak yang dapat digunakan oleh komputer - komputer yang terhubung ke dalam *network*. Komputer - komputer yang terhubung ke dalam jaringan itu biasanya disebut dengan *workstation.* Biasanya kemampuan *workstation* lebih di bawah dari *File server* dan mempunyai aplikasi lain di dalam *harddisk*nya selain aplikasi untuk jaringan. Kebanyakan LAN menggunakan media kabel untuk menghubungkan antara satu komputer dengan komputer lainnya.

b. *Metropolitan Area Network* (MAN) / Jaringan Area Metropolitan

Sebuah MAN, biasanya meliputi area yang lebih besar dari LAN, misalnya antar wilayah dalam satu propinsi. Dalam hal ini jaringan menghubungkan beberapa buah jaringan-jaringan kecil ke dalam lingkungan area yang lebih besar, sebagai contoh yaitu : jaringan Bank dimana beberapa kantor cabang sebuah Bank di dalam sebuah kota besar dihubungkan antara satu dengan lainnya.

c. *Wide Area Network* (WAN) / Jaringan Area Skala Besar

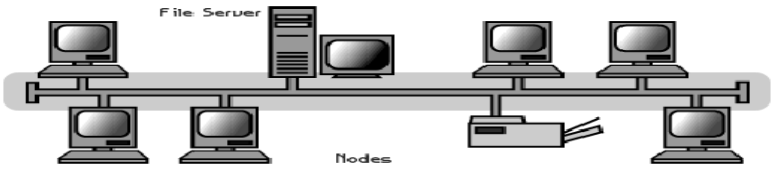
Wide Area Networks (WAN) adalah jaringan komputer yang mencakup area yang besar sebagai contoh yaitu jaringan komputer antar wilayah, kota atau bahkan negara, atau dapat didefinisikan juga sebagai jaringan komputer yang membutuhkan *router* dan saluran komunikasi publik. Biasanya WAN agak rumit dan sangat kompleks, menggunakan banyak sarana untuk menghubungkan antara LAN dan WAN ke dalam komunikasi *global* seperti Internet. Tapi bagaimanapun juga antara LAN, MAN dan WAN tidak banyak berbeda dalam beberapa hal, hanya lingkup areanya saja yang berbeda satu di antara yang lainnya.

2.8.2 Topologi

Topologi adalah suatu cara menghubungkan komputer yang satu dengan komputer lainnya sehingga membentuk jaringan. Cara yang saat ini banyak digunakan adalah *bus, token-ring, star,* dan *tree.* Masing-masing topologi ini mempunyai ciri khas, dengan kelebihan dan kekurangannya sendiri.

1. Topologi BUS

Pada topologi Bus digunakan sebuah kabel tunggal atau kabel pusat di mana seluruh *workstation* dan *server* dihubungkan. Dengan menggunakan *T-Connector* (dengan terminator 50 ohm pada ujung *network*), maka komputer atau perangkat jaringan lainnya bisa dengan mudah dihubungkan satu sama lain.



Gambar 2.2 Topologi Bus

Spesifikasi :

- *Backbone*
- *Coaxial thin / thick*
- Tbase 2 / Tbase 5
- RG-58 / RG- 45
- T-*Connection* / MAU
- Terminator 50 *ohm*
- Jarak antar *Workstation* tidak lebih 2,5 m
- Tiap *Workstation* harus diinstalkan NIC (*Network* Interface Card)

Kelebihan:

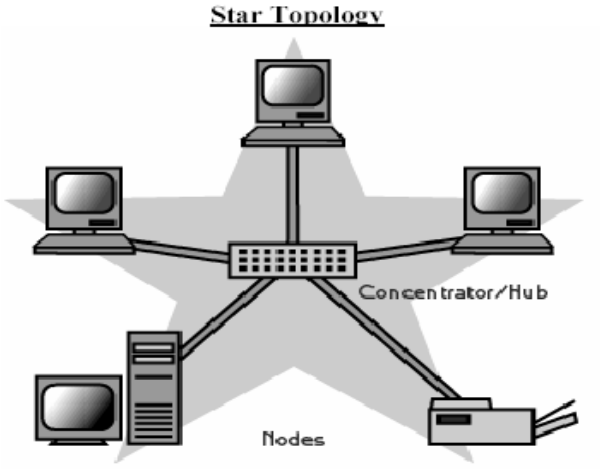
- Mudah dalam instalasi (Pemasangan)
- Sedikit dalam pengunaan kabel
- Biaya relatif murah
- Peripheral Sederhana / sudah banyak

Kekurangan :

- Apabila *backbone* rusak maka jaringan akan lumpuh total
- Apabila ada *Workstation* yang rusak maka jaringan akan lumpuh total
- Sulit mendeteksi kerusakan
- Disarankan untuk tidak dipasang pada area luas
- Bisa mengakibatkan tabrakan data pada *backbone* / *collision*

1.Topologi Star

Pada topologi *Star*, masing-masing *workstation* dihubungkan secara langsung ke *server* atau *hub*.



Gambar 2.3 Topologi Star

Spesifikasi :

- Konsentrator HUB atau Switch
- Kabel UTP / STP tipe 3 atau tipe 5

- Jarak antar tiap segmen node maksimal 100 m
- NIC IEEE 802.3
- Konektor RJ-45

Kelebihan :

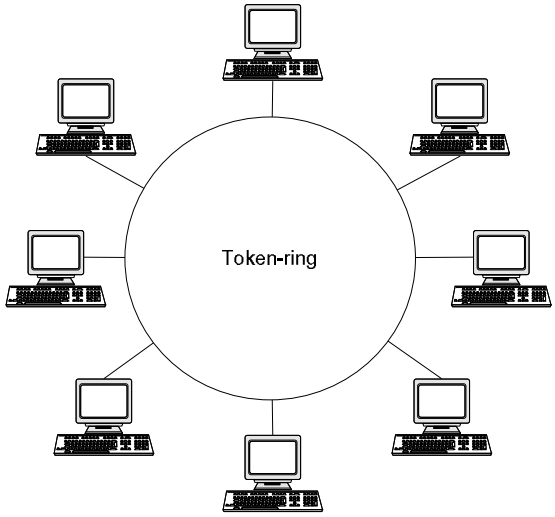
- Paling fleksibel
- Mudah dalam instalasi / pemasangan
- Mudah dalam mendeteksi kerusakan
- Tidak mengalami gangguan apabila memindahkan atau menambahkan *Workstation*

Kekurangan :

- Biaya relatif mahal akibat adanya concentrator
- Jaringan lumpuh apabila concentrator rusak
- Penggunaan kabel relatif banyak

2. Topologi Ring

Topologi ini memanfaatkan kurva tertutup, artinya informasi dan data serta *traffic* disalurkan sedemikian rupa ke masing-masing *node*. Umumnya fasilitas ini memanfaatkan *fiber optic* sebagai sarananya (walaupun ada juga yang menggunakan *twisted pair*). Di dalam topologi Ring semua *workstation* dan *server* dihubungkan sehingga terbentuk suatu pola lingkaran atau cincin. Tiap *workstation* ataupun *server* akan menerima dan melewatkan informasi dari satu komputer ke komputer lain, bila alamat - alamat yang dimaksud sesuai maka informasi diterima dan bila tidak, informasi akan dilewatkan.



Gambar 2.4 Topologi Ring

Spesifikasi :

- Kabel *fiber optik tipe* 4
- *Backbone* membentuk cincin
- HUB 10 Base F
- NIC 10 Base F
- 2 jalur kabel data (*transmit* dan *receive*)
- Satu arah pergerakan data

Kelebihan :

- Tidak adanya tabrakan dalam pengiriman data
- Hemat pemakaian kabel

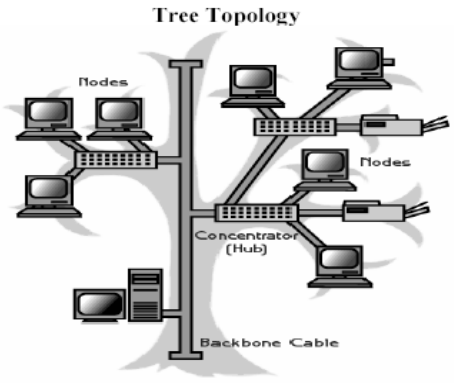
Kekurangan :

- Transmit relatif lama
- Biaya yang dikeluarkan mahal

- Apabila salah satu *workstation* mati maka jaringan lumpuh

3. Topologi *Tree*

Merupakan penggabungan antara dua atau tiga topologi dengan menggunakan suatu alat yang dinamakan *Bridge.*



Gambar 2.5 Topologi Tree

Jaringan dibedakan menjadi dua berdasarkan tipe jaringannya, yaitu tipe *client-server* dan tipe jaringan *peer to peer.*

1. Jaringan *Client-server*

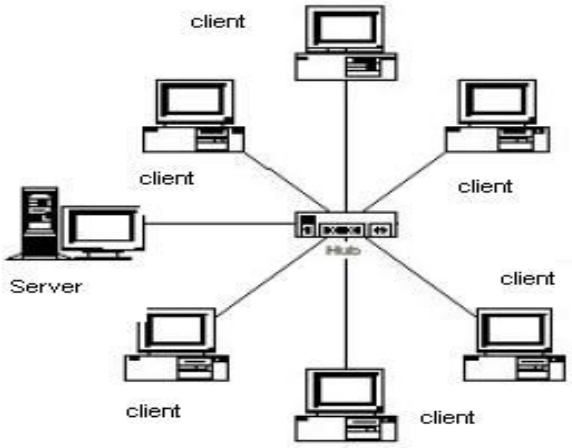
Server adalah komputer yang menyediakan fasilitas bagi komputer-komputer lain di dalam jaringan dan *client* adalah komputer-komputer yang menerima atau menggunakan fasilitas yang disediakan oleh *server*. *Server* pada jaringan tipe *client-server* disebut dengan *Dedicated Server* karena murni berperan sebagai *server* yang menyediakan fasilitas kepada *workstation* dan *server* tersebut tidak dapat berperan sebagai *workstation.*

a. Keunggulan

1. Kecepatan akses lebih tinggi karena penyediaan fasilitas jaringan dan pengelolaannya dilakukan secara khusus oleh satu komputer (*server*) yang tidak dibebani dengan tugas lain sebagai *workstation*.
2. Sistem keamanan dan administrasi jaringan lebih baik, karena terdapat seorang pemakai yang bertugas sebagai administrator jaringan, yang mengelola administrasi dan sistem keamanan jaringan.
3. Sistem *backup* data lebih baik, karena pada jaringan *client-server backup* dilakukan terpusat di *server*, yang akan mem*backup* seluruh data yang digunakan di dalam jaringan

b. Kelemahan

1. Biaya operasional relatif lebih mahal.
2. Diperlukan adanya satu komputer khusus yang berkemampuan lebih untuk ditugaskan sebagai *server*.
3. Kelangsungan jaringan sangat tergantung pada *server*. Bila *server* mengalami gangguan maka secara keseluruhan jaringan akan terganggu.



Gambar 2.6 Jaringan *Client-server*

2. Jaringan *Peer to peer*

Bila ditinjau dari peran *server* di kedua tipe jaringan tersebut, maka *server* di jaringan tipe *peer to peer* diistilahkan *non-dedicated server*, karena *server* tidak berperan sebagai *server* murni melainkan sekaligus dapat berperan sebagai *workstation.*

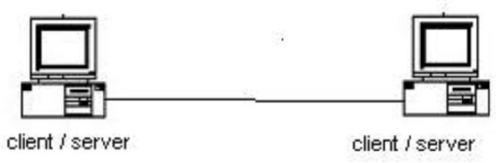
a. Keunggulan

1. Antar komputer dalam jaringan dapat saling berbagi-pakai fasilitas yang dimilikinya seperti: *harddisk, drive,* fax / modem, *printer*.
2. Biaya operasional relatif lebih murah dibandingkan dengan tipe jaringan *client-server*, salah satunya karena tidak memerlukan adanya *server* yang memiliki kemampuan khusus untuk mengorganisasikan dan menyediakan fasilitas jaringan.

3. Kelangsungan kerja jaringan tidak tergantung pada satu *server*. Sehingga bila salah satu komputer/*peer* mati atau rusak, jaringan secara keseluruhan tidak akan mengalami gangguan.

b. Kelemahan

1. *Troubleshooting* jaringan relatif lebih sulit, karena pada jaringan tipe *peer to peer* setiap komputer dimungkinkan untuk terlibat dalam komunikasi yang ada. Di jaringan *client-server*, komunikasi adalah antara *server* dengan *workstation.*
2. Unjuk kerja lebih rendah dibandingkan dengan jaringan *client-server*, karena setiap komputer/*peer* disamping harus mengelola pemakaian fasilitas jaringan juga harus mengelola pekerjaan atau aplikasi sendiri.
3. Sistem keamanan jaringan ditentukan oleh masing-masing *user* dengan mengatur keamanan masing-masing fasilitas yang dimiliki.
4. Karena data jaringan tersebar di masing-masing komputer dalam jaringan, maka *backup* harus dilakukan oleh masing-masing komputer tersebut.



Gambar 2.7 Jaringan *peer to peer*

2.8.3 Perangkat Jaringan Komputer

Perangkat yang umum digunakan dalam membangun sebuah jaringan, adalah sebagai berikut :

1. *File Server*s

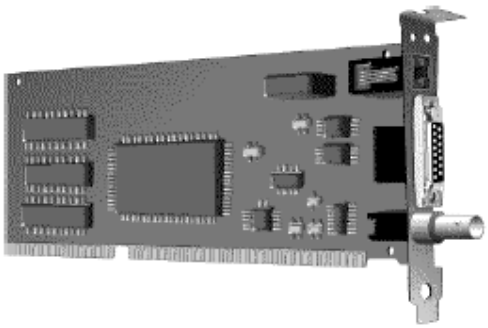
Sebuah *File server* merupakan jantungnya jaringan, merupakan komputer yang sangat cepat, mempunyai memori yang besar, *harddisk* yang memiliki kapasitas besar, dengan kartu jaringan yang cepat. Sistem operasi jaringan tersimpan disini, juga termasuk didalamnya beberapa aplikasi dan data yang dibutuhkan untuk jaringan. Sebuah *File server* bertugas mengontrol komunikasi dan informasi diantara *node*/komponen dalam suatu jaringan. Sebagai contoh mengelola pengiriman *File* database atau pengolah kata dari *workstation* atau salah satu *node*, ke *node* yang lain, atau menerima *e-mail* pada saat yang bersamaan dengan tugas yang lain terlihat bahwa tugas *File server* sangat kompleks, *File server* juga harus menyimpan informasi dan membaginya secara cepat

2. *Workstation*s

Keseluruhan komputer yang terhubung ke *File server* dalam jaringan disebut sebagai *workstation*. Sebuah *workstation* minimal mempunyai kartu jaringan, aplikasi jaringan (*software* jaringan), kabel untuk menghubungkan ke jaringan, biasanya sebuah *workstation* tidak begitu membutuhkan *Floppy Disk* karena data yang ingin di simpan dapat diletakkan di *File server*. Hampir semua jenis komputer dapat digunakan sebagai komputer *workstation*.

3. *Network Interface Cards* (NIC) atau *Ethernet* card

Kartu jaringan *Ethernet* biasanya dibeli terpisah dengan komputer, kecuali untuk beberapa Motherboard yang sudah terintegrasi *Ethernet* card dengan konektor RJ-45. Kartu Jaringan *ethernet* umumnya telah menyediakan port koneksi untuk kabel *Koaksial* ataupun kabel *twisted pair*, jika didesain untuk kabel *koaksial* konektornya adalah BNC, dan apabila didesain untuk kabel *twisted pair* maka konektornya RJ-45.



Gambar 2. 8 *Ethernet Card*

4. Pengkabelan

Terdapat beberapa tipe pengkabelan yang biasa digunakan dan dapat digunakan untuk mengaplikasikan jaringan, yaitu:

a. *Thin Ethernet (Thinnet)*

Thin Ethernet atau *Thinnet* memiliki keunggulan dalam hal biaya yang relatif lebih murah dibandingkan dengan tipe pengkabelan lain, serta pemasangan komponennya lebih mudah. Panjang kabel *thin coaxial* / RG-58 antara 0.5 – 185 m dan maksimum 30 komputer terhubung.

b. *Thick Ethernet (Thicknet)*

Dengan *thick Ethernet* atau *thicknet,* jumlah komputer yang dapat dihubungkan dalam jaringan akan lebih banyak dan jarak antara komputer dapat diperbesar, tetapi biaya pengadaan pengkabelan ini lebih mahal serta pemasangannya relatif lebih sulit dibandingkan dengan *Thinnet*. Pada *Thicknet* digunakan *transceiver* untuk menghubungkan setiap komputer dengan sistem jaringan dan konektor yang digunakan adalah konektor *tipe* DIX. Panjang kabel *transceiver* maksimum 50 m, panjang kabel *Thick Ethernet* maksimum 500 m dengan maksimum 100 *transceiver* terhubung.



Gambar 2.9 Kabel *Thicknet* dan *Thinnet*

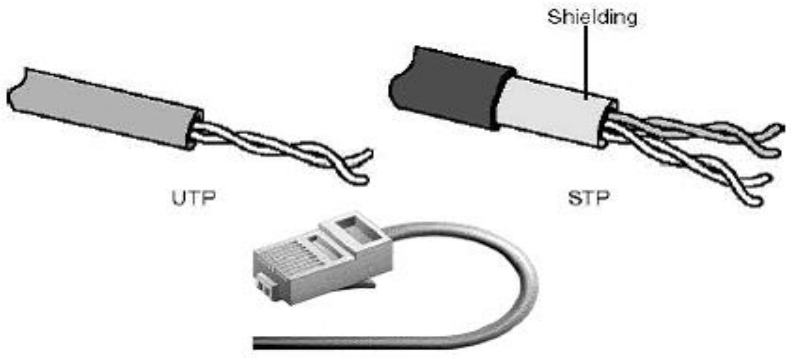
c. *Twisted pair*

Kabel *Twisted pair* ini terbagi menjadi dua jenis yaitu STP (*Shielded Twisted pair*) dan UTP (*Unshielded.Twisted pair*). *Shielded* adalah jenis kabel yang memiliki selubung pembungkus sedangkan *unshielded* tidak mempunyai selubung pembungkus. Untuk koneksinya kabel jenis ini menggunakan konektor RJ-11 atau RJ-45. Pada *twisted pair* (10 BaseT) *network*, komputer disusun membentuk suatu pola star. Setiap PC memiliki satu kabel *twisted pair* yang tersentral pada HUB. *Twisted pair* umumnya lebih handal (*reliable*) dibandingkan dengan thin coax karena HUB mempunyai kemampuan data *error correction* dan

meningkatkan kecepatan transmisi. Saat ini ada beberapa *grade*, atau kategori dari kabel *twisted pair*. Kategori 5 adalah yang paling *reliable* dan memiliki kompabilitas yang tinggi, dan yang paling disarankan. Berjalan baik pada 10Mbps dan Fast *Ethernet* (100Mbps). Kabel kategori 5 dapat dibuat *straight-thro*ugh atau *crossed*. Kabel *straight through* digunakan untuk menghubungkan komputer ke HUB. Kabel *crossed* digunakan untuk menghubungkan HUB ke HUB. Panjang kabel maksimum kabel *Twisted-Pair* adalah 100 m.

Tabel 2.1 Kategori Kabel *twisted pair*

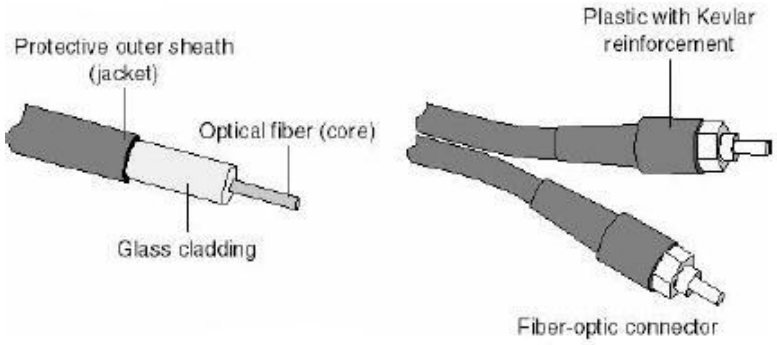
Type	***Use***
1	*Voice only (Phone wire*)
2	Data to 4 MBps (*local talk*)
3	Data to 10 MBps (LAN *ethernet*)
4	Data to 16 – 20 MBps (LAN topologi Ring)
5	Data to 100 MBps (Fast *ethernet*)



Gambar 2.10 Kabel UTP, STP dan Konektor RJ-45

d. *Fibre Optic*

Jaringan yang menggunakan *Fibre Optic* (FO) biasanya perusahaan besar, dikarenakan harga dan proses pemasangannya lebih sulit. Namun demikian, jaringan yang menggunakan *Fibre Optic* dilihat dari segi kehandalan dan kecepatan tidak diragukan. Kecepatan pengiriman data dengan media *Fibre Optic* lebih dari 100Mbps dan bebas pengaruh lingkungan.



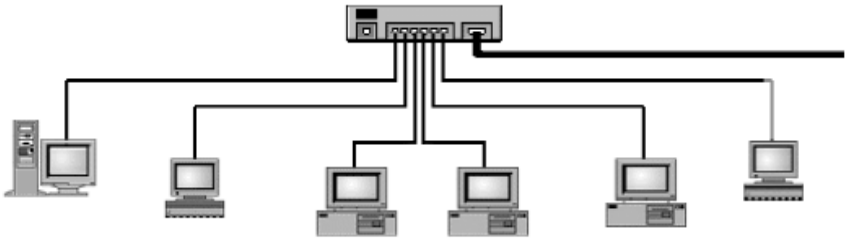
Gambar 2.11 Kabel Fiber Optik

5. Hub / Konsentrator

Sebuah Konsentrator/Hub adalah sebuah perangkat yang menyatukan kabel-kabel jaringan dari tiap - tiap *workstation, server* atau perangkat lain. Dalam topologi *Star*, kabel *twisted pair* berasal dari sebuah *workstation* terhubung ke Hub. Hub mempunyai banyak slot concentrator yang dapat dipasang menurut nomor *port* dari *card* yang dituju.

Ciri-ciri yang dimiliki Konsentrator adalah :

a. Terdiri dari 8, 12, atau 24 *port* RJ-45
b. Digunakan pada topologi Bintang / *Star*
c. Di jual dengan aplikasi khusus yaitu aplikasi yang mengatur manajemen port tersebut.
d. Biasanya disebut hub
e. Di pasang pada rak khusus, yang didalamnya ada *Bridge*s, router



Gambar 2.12 Hub atau Konsentrator

6. *Repeater*

Repeater adalah alat untuk memperkuat sinyal dalam jaringan. Contoh pemakaiannya yang paling mudah adalah pada sebuah LAN menggunakan topologi *Star* dengan menggunakan kabel UTP. Dimana diketahui panjang maksimal untuk sebuah kabel UTP adalah 100 meter, maka untuk menguatkan sinyal dari kabel tersebut dipasanglah sebuah repeater pada jaringan tersebut.



Gambar 2.13 *Repeater*

7. *Bridge*s

Bridge adalah sebuah perangkat yang membagi satu buah jaringan ke dalam dua buah jaringan, ini digunakan untuk mendapatkan jaringan yang efisien, dimana kadang pertumbuhan *network* sangat cepat oleh karena itu diperlukan jembatan untuk menghubungkan beberapa *network*. Kebanyakan *Bridges* dapat mengetahui masing-masing alamat dari tiap-tiap segmen komputer pada jaringan sebelahnya dan juga pada jaringan yang lain di sebelahnya pula. Diibaratkan bahwa *Bridges* ini seperti polisi lalu lintas yang mengatur di persimpangan jalan pada saat jam-jam sibuk. *Bridge* mengatur agar informasi di antara kedua sisi *network* tetap berfungsi dengan baik dan teratur. *Bridges* juga dapat di gunakan untuk mengkoneksi di antara *network* yang menggunakan *tipe* kabel yang berbeda ataupun topologi yang berbeda pula.

8. *Routers*

Sebuah *Router* menerjemahkan informasi dari satu jaringan ke jaringan yang lain, *router* hampir sama dengan *Bridge* namun agak pintar sedikit, *rout*er

akan mencari jalur yang terbaik untuk mengirimkan sebuah pesan yang berdasarkan atas alamat tujuan dan alamat asal. Sementara *Bridges* dapat mengetahui alamat masing-masing komputer di masing -masing sisi jaringan, *router* mengetahui alamat komputer, *bridges* dan *router* lainnya. *Router* dapat mengetahui keseluruhan jaringan, mengontrol sisi mana yang paling sibuk dan bisa menarik data dari sisi yang sibuk tersebut sampai sisi tersebut bersih. Sebuah router dapat menerjemahkan informasi antara LAN dan Internet, artinya *router* mencarikan alternatif jalur yang terbaik untuk mengirimkan data melewati internet.

Fungsi *Router* :

a. Mengatur jalur sinyal secara *effisien*
b. Mengatur pesan di antara dua buah protocol
c. Mengatur pesan di antara topologi jaringan *linear* Bus dan Bintang / *Star*
d. Mengatur pesan antara kabel *Fibre optic*, kabel *koaksial* atau kabel *twisted pair*.

9. Protokol TCP / IP

Protokol TCP/IP merupakan teknologi "*Packet Switching*" yang berasal dari proyek DARPA (*Development of Defense Advanced Research Project Agency*) di tahun 1970-an yang dikenal dengan nama ARPANET.

TCP/IP adalah protokol yang tersedia pada NT 4.0 dengan layanan aplikasi berorientasi intranet dan internet. TCP/IP sendiri sebenarnya merupakan tempat dari gabungan beberapa protokol. Di dalamnya terdapat protokol TCP, IP, SMTP, POP, dan sebagainya.

TCP (*Transmission Control Protocol*) melakukan transmisi data per segmen, artinya paket data dipecah dalam jumlah yang sesuai dengan besaran paket, kemudian dikirim satu persatu hingga selesai. Agar pengiriman data sampai dengan baik, maka pada setiap paket pengiriman, TCP akan menyertakan nomor seri (*sequence number*). Komputer mitra yang menerima paket tersebut harus mengirim balik sebuah sinyal *ACKnowledge* dalam satu periode yang ditentukan. Bila pada waktunya sang mitra belum juga memberikan ACK, maka terjadi "*time out*" yang menandakan pegiriman paket gagal dan harus diulang kembali. Model protocol TCP disebut sebagai *connection oriented protocol.*

Karena penting peranannya pada sistem operasi *Windows* dan juga karena *protokol* TCP/IP merupakan *protokol* pilihan (*default*) dari *Windows. Protokol* TCP berada pada lapisan *Transport* model OSI (*Open* Sistem *Interconnection*), sedangkan IP (*Internet Protocol*) berada pada lapisan *Network* mode OSI.

2.8.4 IP Address

IP *address* adalah alamat yang diberikan pada jaringan komputer dan peralatan jaringan yang menggunakan protokol TCP/IP. IP *address* terdiri atas 32 bit angka biner yang dapat dituliskan sebagai empat kelompok angka desimal yang dipisahkan oleh tanda titik seperti 192.168.0.1.

Tabel 2.2 *Contoh IP address*

***Network* ID**			***Host* ID**
192	168	0	1

IP *address* terdiri atas dua bagian yaitu *network* ID dan *host* ID, dimana *network* ID menentukan alamat jaringan komputer, sedangkan *host* ID menentukan alamat *host* (komputer, *router, switch*). Oleh sebab itu IP *address* memberikan alamat lengkap suatu *host* beserta alamat jaringan di mana *host* itu berada.

Untuk mempermudah pemakaian, bergantung pada kebutuhan pemakai, IP *address* dibagi dalam tiga kelas seperti diperlihatkan pada tabel 2.3.

Tabel 2.3 Kelas IP *Address*

Kelas	***Network* ID**	***Host* ID**	***Default Subnet Mask***
A	xxx.0.0.1	xxx.255.255.254	255.0.0.0
B	xxx.xxx.0.1	xxx.xxx.255.254	255.255.0.0
C	xxx.xxx.xxx.1	xxx.xxx.xxx.254	255.255.255.0

IP *address* kelas A diberikan untuk jaringan dengan jumlah *host* yang sangat besar. Range IP 1.xxx.xxx.xxx. – 126.xxx.xxx.xxx, terdapat 16.777.214 (16 juta) IP *address* pada tiap kelas A. IP *address* kelas A diberikan untuk jaringan dengan jumlah *host* yang sangat besar. Pada IP *address* kelas A, *network*

ID ialah 8 bit pertama, sedangkan *host* ID ialah 24 bit berikutnya. Dengan demikian, cara membaca IP *address* kelas A, misalnya 113.46.5.6 ialah: *Network* ID = 113, *Host* ID = 46.5.6 Sehingga IP *address* diatas berarti *host* nomor 46.5.6 pada *network* nomor 113. IP *address* kelas B biasanya dialokasikan untuk jaringan berukuran sedang dan besar. Pada IP *address* kelas B, *network* ID ialah 16 bit pertama, sedangkan *host* ID ialah 16 bit berikutnya. Dengan demikian, cara membaca IP *address* kelas B, misalnya 132.92.121.1 *Network* ID = 132.92 *Host* ID = 121.1 Sehingga IP *address* di atas berarti *host* nomor 121.1 pada *network* nomor 132.92. dengan panjang *host* ID 16 bit, *network* dengan IP *address* kelas B dapat menampung sekitar 65000 *host*. Range IP 128.0.xxx.xxx – 191.155.xxx.xxx IP *address* kelas C awalnya digunakan untuk jaringan berukuran kecil (LAN). *Host* ID ialah 8 bit terakhir. Dengan konfigurasi ini, bisa dibentuk sekitar 2 juta *network* dengan masing-masing *network* memiliki 256 IP *address*. Range IP 192.0.0.xxx – 223.255.255.x. Pengalokasian IP *address* pada dasarnya ialah proses memilih *network* ID dan *host* ID yang tepat untuk suatu jaringan. Tepat atau tidaknya konfigurasi ini tergantung dari tujuan yang hendak dicapai, yaitu mengalokasikan IP *address* seefisien mungkin.

2.9 Data Flow Diagram (DFD)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interkasi antara data yang tersimpan dan proses yang dikenakan pada data tersebut.

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan.

DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. Kelebihan utama pendekatan alir data, yaitu :

1. Kejelasan dari menjalankan implementasi teknis sistem.
2. Pemahaman lebih jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
3. Mengkomunikasikan pengetahuan sistem yang ada dengan pengguna melalui diagram alir data.
4. Menganalisis sistem yang diajukan untuk menentukan apakah data-data dan proses yang diperlukan sudah ditetapkan.

DFD terdiri dari *context diagram* dan diagram rinci (DFD *leveled*), *context diagram* berfungsi memetakan model lingkungan (menggambarkan hubungan antara entitas luar, masukan dan keluaran sistem), yang direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem. DFD *leveled* menggambarkan sistem jaringan kerja antara fungsi yang berhubungan satu sama lain dengan aliran data penyimpanan data, model ini hanya memodelkan sistem dari sudut pandang fungsi.

2.10 Delphi 7.0

Bahasa pemrograman *Delphi* merupakan pemrograman *visual* (berbasis windows) yang dibuat oleh sebuah Perusahaan *Software Borland .Inc.* Bahasa

pemrograman *Delphi* merupakan bahasa pemrograman yang dikembangkan dari bahasa pemrograman *Pascal*. Fungsi dari aplikasi ini sama dengan fungsi aplikasi *visual* lainnya.

Delphi 7.0 dapat menangani pembuatan aplikasi sederhana hingga aplikasi berbasis jaringan. *Delphi* 7.0 dapat dimanfaatkan untuk membuat aplikasi berbasis teks, *grafis*, angka, *database* maupun web.

Bahasa pemrograman visual memiliki dua hal yaitu *object* dan kode program. Objek berbentuk komponen yang dapat dilihat (*visual*), sedangkan kode program merupakan sekumpulan teks yang digunakan sebagai perintah yang telah diatur dengan suatu aturan.