BAB II

LANDASAN TEORI

2.1 Kompresi Data

Sistem komunikasi dirancang untuk mentransmisi informasi yang dibangkitkan oleh sumber ke beberapa tujuan. Sumber informasi mempunyai beberapa bentuk yang berbeda. Sebagai contoh, dalam radio *broadcasting*, sumber biasanya sumber audio (suara atau music). dalam TV *broadcasting*, sumber informasi biasanya sebuah sumber video yang keluarannya berupa *image* bergerak. Output dari sumber-sumber ini adalah sinyal analog dan sumbernya disebut sumber analog.

Kontras dengan komputer dan tempat penyimpanan data (*storage*) seperti disk magnetic atau optical, menghasilkaan ouput berupa sinyal diskrit (biasanya karakter binary atau ASCII). Sinyal yang menjadi sumbernya biasa disebut dengan sumber diskrit. baik sumber analog maupun diskrit, sebuah komunikasi digital dirancang untuk mentransmisi informasi dalam bentuk digital. Sehingga konsenkuensi keluaran dari sumber harus diubah dahulu menjadi bentuk keluaran sumber digital yang biasanya dilakukan pada *source encoder*. Keluaran dari sumber tersebut dapat diasumsikan menjadi sebuah digit biner sekuensial.

Pada akhir tahun 40-an dimana dimulainya tahun teori informasi, ide pengembangan metode coding yang efesien baru dimulai dan dikembangkan. Dimulainya penjelajahan ide dari *entropy*, *information content* dan *redudansi*.

Salah satu ide yang popular adalah apabila probabilitas dari simbol dalam suatu pesan diketahui, maka terdapat cara untuk mengkodekan simbol, sehingga pesan memakan tempat yang lebih kecil. Ide inilah yang menjadi cikal bakal dalam terciptanya kompresi data.

Model pertama yang muncul unutk kompresi sinyal digital adalah Shannon-Fano coding. Shannon dan fano (1948) terus menerus mengembangkan algoritma ini yang menghasilkan codeord biner untuk setiap symbol (unik) yang terdapat pada data file.

Huffman coding (1952) memakai hampir semua karakteristik dari *Shannon-fano coding*. Huffman coding dapat menghasilkan kompresi data yang efektif dengan mengurangkan jumlah redudansi dalam mengkodingkan simbol. Telah dapat dibuktikan, bahwa Huffman coding merupakan metode *fixed-length* yang paling efesien.

Pada limabelas tahun terakhir, Huffman coding telah digantikan oleh *Arithmetic coding*. Arithmetic coding melewatkan ide untuk menggantikan sebuah simbol masukan dengan kode yang spesifik. Algoritma ini menggantikan sebuah aliran simbol masukan dengan sebuah angka keluaran *single floating-point*. Lebih banyak bit dibutuhkan dalam angka keluaran, maka semakin rumit pesan yang diterima.

Algoritma dictionary-based compression menggunakan metode yang sangat berbeda dalam mengkompres data. Algoritma ini menggantikan string variable-length dari simbol menjadi sebuah *token.* Token merupakan sebuah indek dalam

susunan kata di kamus. Apabila token kecil dari susunan kata, maka token akan menggantikan prase tersebut dan kompresi pun terjadi.

Kompresi data (pemampatan data) merupakan suatu teknik untuk memperkecil jumlah ukuran data (hasil kompresi) dari data aslinya. Pemampatan data umumnya diterapkan pada mesin komputer, hal ini dilakukan karena setiap simbol yang dimuncul pada komputer memiliki nilai bit-bit yang berbeda. Misal pada ASCII setiap simbol yang dimunculkan memiliki panjang 8 bit, misal kode A pada ASCII mempunyai nilai decimal 65, jika dirubah dalam bilangan biner menjadi 010000001. Pemampatan data digunakan untuk mengurangkan jumlah bir-bit yang dihasilkan dari setiap simbol yang muncul. Dengan pemampatan ini diharapkan dapat mengurangi (memperkecil ukuran data) dalam ruang penyimpanan.

2.1.1 Faktor Penting Kompresi Data

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

1. *Time process*

   Yaitu waktu yang dibutuhkan untuk menjalankan proses kompresi.

2. *Completeness*

   Yaitu kelengkapan data setelah data tersebut dikompresi.

3. *Ratio Compress*

   Yaitu ukuran data setelah data tersebut dikompresi

4. *Optimaly*

   Yaitu perbandingan ukuran data sebelum dikompresi dengan data yang telah dikompresi.

2.1.2 Jenis Kompresi Data

Berdasarkan mode penerimaan data oleh manusia, kompresi data dapat dibagi menjadi dua, yaitu :

a. *Dialoque Mode*

   yaitu proses penerimaan data dimana pengirim dan penerima seakan berdialog (*real time*), dimana kompresi data harus berada dalam batas penglihatan dan pendengaran mausia. Contohnya pada *video conference*.

b. *Retrieval Mode*

   yaitu proses penerimaan data tidak dilakukan secara real time. Jenis kompresi ini dapat dilakukan *random access* terhadap data dan dapat bersifat interaktif.

2.1.3 Metode Kompresi Data

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi file input) menjadi sekumpulan *codeword*, metode kompresi terbagi menjadi dua kelompok, yaitu :

a. Metode Static

   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. Contohnya pada *Huffman static, arithmetic coding*

b. Metode Dinamik (adaptif)

Menggunakan peta kode yang dapat diubah 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 diperlakukan satu kali pembacaan terhadap isi file. Contohnya pada algoritma LZW dan DMC.

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

a. *Metode simbolwise*

Menghitung peluang kemunculan dari tiap simbol dalam file input, lalu mengkodekan satu simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek dibandinglan simbol yang lebih jarang muncul. Contohnya pada Huffman coding, arithmetic coding.

b. *Metode dictionary*

Menggantikan karakter/fragmen dalam file input dengan indeks lokasi dari karakter/fragmen tersebut dalam sebuah kamus (*dictionary*). Contohnya pada algoritma LZW

c. *Metode predictive*

Menggunakan model *finite-context* atau *finite-state* untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya. Contohnya pada algoritma DMC

2.1.4 Teknik Kompresi Data

Teknik kompresi data dapat digolongkan menjadi dua kelompok utama yaitu : lossy dan lossless. Teknik kompresi secara lossy dimaksudkan dengan teknik kompresi data dengan menghilangkan ketelitian data utama guna mendapatkan data sekecil mungkin (kompresi data sebesar mengkin). Teknik kompresi data secara lossless yaitu teknik kompresi data dengan mengurangkan jumlah data yang terjadi redudansi (memiliki symbol yang sama) sebelum terjadi kompresi.

a. *Lossy Compression*

Merupakan teknik kompresi yang menghilangkan beberapa informasi data yang dianggap tidak penting. Sehingga hasil data yang telah terkompresi tidak sama dengan data yang sebelum dikompresi. Namun data yang telah terkompresi tersebut sudah cukup untuk digunakan, walaupun datanya telah berubah. Keunggulan dari teknik ini adalah file atau data yang telah terkompresi memiliki ukuran yang lebih kecil dari ukuran data aslinya. Biasanya teknik kompresi ini banyak diaplikasikan pada data gambar dan data audio.

b. *Lossless Compression*

Merupakan teknik kompresi yang mempertahankan kebutuhan informasi yang dikandung oleh data, sehingga informasi yang terkandung pada file yang telah terkompresi tetap terjaga meskipun ukurannya telah berubah dari ukuran data aslinya. Keunggulan dari teknik ini adalah data yang telah terkompresi, apabila didekompresi kembali akan menghasilkan data yang sama persis dengan data aslinya. Biasanya teknik kompresi ini banyak diaplikasikan pada data teks.

2.1.5 Klasifikasi Teknik Kompresi

Teknik kompresi dapat diklasifikasikan menjadi tiga kelompok, antara lain :

a. *Entropy Encoding*

Merupakan klasifikasi teknik kompresi yang bersifat *loseless compression.* Tekniknya tidak berdasarkan media dengan spesifikasi dan karakteristik tertentu, namun berdasarkan urutan data. Dan *statistical encoding* pada klasifikasi teknik ini tidak memperhatikan semantik data. Contoh kelompok teknik kompresi ini adalah pada *Run-Length* coding, *Huffman* coding, *Arithmetic* coding.

b. *Source Coding*

Merupakan klasifikasi teknik kompresi yang bersifat *lossy compression.* Tekniknya berkaitan dengan data semantik (arti data) dan media. Contoh kelompok teknik kompresi ini adalah pada *prediction (DPCM, DM), transformation (FFT, DCT), layered coding (bit position, subsampling, sub-band coding), vector quantization*

c. *Hybrid Coding*

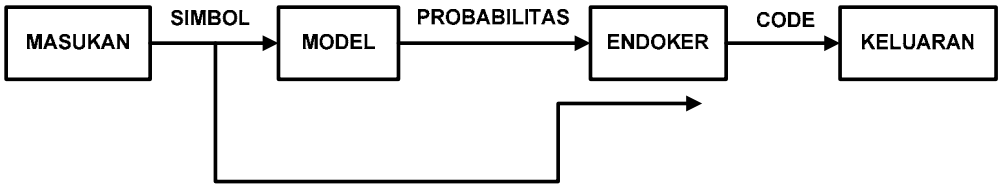
Merupakan klasifikasi teknik kompresi yang bersifat gabungan antara *lossy compression* dan *loseless compression.* Contoh kelompok teknik kompresi ini adalah pada JPEG, MPEG, H.261, DVI

2.1.6 Encoding dan Decoding

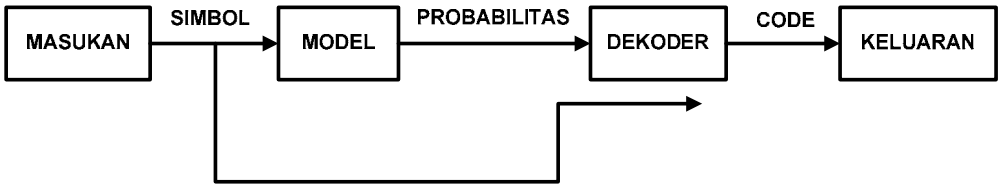
Encoding merupakan teknik untuk mendapatkan kode-kode tertentu (encoder). Dari kode-kode tersebut dapat diaplikasikan untuk pemampatan data dan keamanan data. Dari data-data yang telah dikodekan tersebut, format-format isi dari data tersebut berbentuk kode-kode yang tidak bisa dibaca oleh user. Agar kode-kode tersebut bisa dibaca oleh user, maka kita perlu mengkodekan ulang data tersebut. Hal ini biasa dikenal dengan nama *decoding* (decoder)

Secara umum pemampatan data merupakan merubah suatu simbol-simbol menjadi suatu kode-kode. Pemampatan dikatakan efektif jika ukuran perolehan kode-kode tersebut sangat kecil dibandingkan dengan ukuran kode simbol aslinya. Dari suatu kode-kode atau simbol-simbol dasar suatu model akan dinyatakan dalam kode khusus.

Proses decoding, yaitu proses pengembalian kode-kode yang telah dibuat menjadi simbol-simbol yang dikenal oleh user. Proses decoder ini membaca *header* dari kode-kode yang berisi informasi simbol dan jumlah simbol yang digunakan, setelah pembacaan header proses encoder dengan model ditunjukan pada gambar 2.1, sedangkan proses decoder dengan model ditunjukan pada gambar 2.2 dibawah ini



Gambar 2.1 Model Proses Encoder



Gambar 2.2 Model Proses Dekoder

2.2 Pengkodean Data

Data disimpan di dalam komputer pada *main memory* untuk diproses. Sebuah karakter data disimpan dalam *main memory* menempati posisi 1 byte. Komputer generasi pertama, 1 byte terdiri dari 4 bit, komputer generasi kedua, 1 byte terdiri dari 6 bit dan komputer generasi sekarang, 1 byte terdiri dari 8 bit. Suatu karakter data yang disimpan di *main memory* diwakili dengan kombinasi dari digit binary (binary digit atau bit). Suatu kode biner dapat digunakan untuk mewakili suatu karakter.

Suatu komputer yang berbeda menggunakan kode biner yang berbeda untuk mewakili suatu karakter. Komputer yang 1 byte terdiri dari 4 bit, menggunakan kode binary yang berbentuk kombinasi 4 bit, yaitu BCD (*Binary coded decimal*). Komputer yang menggunakan 6 bit untuk 1 bytenya, menggunakan kode biner yang terdiri dari kombinasi 6 bit, yaitu SBCDIC (*Standard Binary Coded Decimal*

Interchange Code). Komputer yang terdiri dari 8 bit, menggunakan kode biner yang terdiri dari kombinasi 8 bit, yaitu EBCDIC (*extended Binary coded decimal interchange code*) atau ASCII (*American standard code of information interchange*).

2.2.1 BCD (Binary Coded Decimal)

BCD merupakan kode biner yang digunakan hanya untuk mewakili nilai desimal saja, yaitu angka 0 sampai dengan 9. BCD menggunakan kombinasi dari 4 bit, sehingga sebanyak 16 (2^4 = 16) kemungkinan kombinasi yang dapat diperoleh dan hanya 10 kombinasi yang dipergunakan.

Kode BCD yang orisinil sudah jarang dipergunakan untuk komputer generasi sekarang, karena tidak dapat mewakili huruf atau simbol-simbol karakter khusus.

2.2.2 SBCDIC (Standard Binary Coded Decimal Interchange Code)

SBCDIC merupakan kode biner perkembangan dari BCD. BCD dianggap tanggung, karena masih ada 6 kombinasi yang tidak dipergunakan, tetapi tidak dapat digunakan untuk mewakili karakter yang lainnya. SBCDIC menggunakan kombinasi 6 bit , sehingga lebih banyak kombinasi yang bisa dihasilkan, sebanyak 64 kombinasi kode, yaitu 10 kode untuk digit angka, 26 kode untuk huruf alphabet dan sisanya karakter-karakter khusus yang dipilih [1].

Posisi bit di SBCDIC dibagi menjadi 2 zone, yaitu 2 bit pertama disebut dengan *alphabet position* dan 4 bit berikutnya disebut dengan *numeric bit position.*

2.2.3 EBCDIC (Extended Binary Coded Decimal Interchange Code)

EBCDIC terdiri dari kombinasi 8 bit yang memungkinkan untuk mewakili karakter sebanyak 256 kombinasi karakter. Pada EBCDIC, *high-order* bits atau 4 bit pertama disebut dengan zone bits dan *low order* bits atau 4 bit kedua disebut dengan numeric bit.

2.2.4 ASCII (American standard code for information interchange) 7 bit

Kode ASCII yang standard menggunakan kombinasi 7 bit, dengan kombinasi sebanyak 127 dari 128 kemungkinan kombinasi.

Kode ASCII 7 bit ini terdiri dari dua bagian, yaitu *control characters* dan *information characters* merupakan karakter-karakter yang mewakili data [1].

2.2.5 ASCII 8 bit

Karakter-karakter graphic yang tidak dapat diwakili oleh ASCII 7 bit, dapat diwakili dengan ASCII 8 bit karena lebiih banyak memberikan kombinasi karakter.

2.3 Komunikasi Data

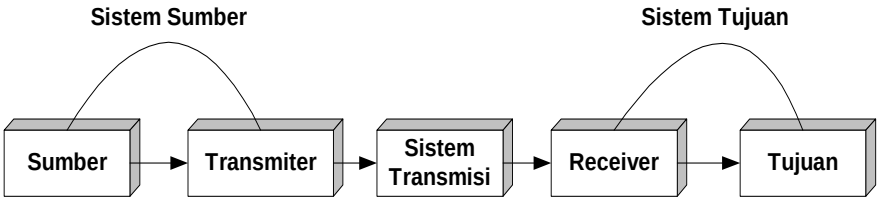
Komunikasi data adalah proses pengiriman informasi diantara dua titik menggunakan kode biner melewati saluran transmisi dan peralatan switching dapat terjadi antara komputer dengan komputer, komputer dengan terminal atau komputer dengan peralatan. Komunikasi data merupakan gabungan dari teknik telekomunikasi dengan teknik pengolahan data.

Adapun tujuan dari komunikasi data adalah sebagai berikut :

1. Memunkinkan pengiriman data dalam jumlah besar efisien, tanpa kesalahan dan ekomis dari suatu tempat ketempat yang lain.
2. Memungkinkan penggunaan sistem komputer dan perlatan pendukung dari jarak jauh (*remote computer use).*
3. Memungkinkan penggunaan komputer secara terpusat maupun secara tersebar sehingga mendukung manajemen dalam hal kontrol, baik desentralisasi ataupu sentralisasi.
4. Mempermudah kemungkinan pengelolaan dan pengaturan data yang ada dalam berbagai mcam sistem komputer.
5. Mengurangi waktu untuk pengelolaan data.
6. Mendapatkan da langsung dari sumbernya.
7. Mempercepat penyebarluasan informasi.

2.3.1 Model Komunikasi Data

Komunikasi data berkaitan dengan pertukaran data diantara dua perangkat yang terhubuang secara langsung yang memungkinkan adanya pertukaran data antar kedua pihak.gambar 2.1 menggambarkan proses komunikasi data.



Gambar 2.3 Komunikasi Data

Pada gambar diatas terdapat elemen-elemen dalam kunci model tersebut :

1. *Source* (sumber) : Alat ini membangkitkan data sehingga dapat ditransmisikan, contoh telepon, *Personal Computer (PC)*
2. *Transmitter* (pengirim): Biasanya data yang dibangkitkan dari sister sumber tidak ditransmisikansecara langsung dalam bentuk aslinya. Sebuah *transmitter* cukup memindah dan menandai informasi dengan cara yang sama seperti sinyal-sinyal elektromagnetik yang dapat ditransmisikan melewati beberapa sistem transmisi berurutan.
3. Sistem transmisi : Berupa jalur transmisi tunggal *(single transmission*)atau jaringan komplek(*complex networ*k)yang menghubungkan antara sumber dengan tujuan (*destination).*
4. Tujuan (*destination)* : menangkap data yang dihasilkan oleh *receiver*

2.3.2 Jaringan Komunikasi Data

Jaringan komunikasi dapat diartikan sebagai suatu sistem yang terbentuk dari interkoneksi fasilitas-fasilitas yang dirancang untuk membawa trafik dari beragam sumber telekomunikasi.

Suatu jaringan terdiri dari *link* dan *node*. Istilah *node* digunakan untuk merepresentasikan sentral, *junction* atau keduanya. Istilah *link* digunakan untuk merepresentasikan kabel, peralatan transmisi, dan sebagainya. Sedangkan trafik adalah informasi yang terdapat di dalam jaringan, yang mengalir melalui *link* dan *node*.

Suatu jaringan komunikasi merupakan sumber daya yang dapat dipakai secara bersamaan (*shared*) oleh sejumlah *end user* untuk berkomunikasi dengan

user lain yang likasinya berjauhan. Tidak semua *user* menggunakan jaringan pada waktu yang bersamaan, oleh karena itu merupakan suatu hal yang logis apabila sumber daya jaringanyang sangat penting ini dipakai bersama-sama. Penggunaan sumber daya secara bersamaan ini melahirkan konsep sentral.

a. LAN (Local Area Network)

LAN digunakan untuk menghubungkan komputer yang berada di dalam suatu area yang kecil, misalnya di dalam suatu gedung perkantoran atau kampus. Jarak antar komputer yang dihubungkan bias mencapai 5 sampai 10 km. Suatu LAN biasnya bekerja pada kecepatan mulai 10 Mbps sampi 100 Mbps. LAN menjadi populer karena memungkinkan banyak pengguna untuk memakai sumber daya yang dapat digunakan itu misalnya suatu *mainframe, file server,* printer, dan sebagainya.

b. MAN (Metropolitan Area Network)

MAN merupakan suatu jaringan yang cakupannya meliputi suatu kota. MAN menghubungkan LAN-LAN yang lokasinya berjauhan. Jangkauan MAN mencapai 10 km sampai beberapa ratus km. Suatu MAN biasanya bekerja pada kecepatan 1,5 sampai 150 Mbps.

c. WAN (Wide Area Network)

WAN dirancang untuk menghubungkan komputer-komputer yang terletak pada suatu cakupan geografis yang luas,seperti hubungan dari suatu kota ke kota yang lain didalm suatu Negara. Cakupan WAN bias meliputi 100 km sampai 1.000 km, dan kecepatan antar kota bias bervariasi antara 1,5 Mbps sampai 2,4

Gbps. Dalam WAN, biaya untuk peralatan untuk transmisi sangat tinggi,dan biasanya jaringan WAN dimiliki dan dioperasikan sebagai suatu jaringan *public*.

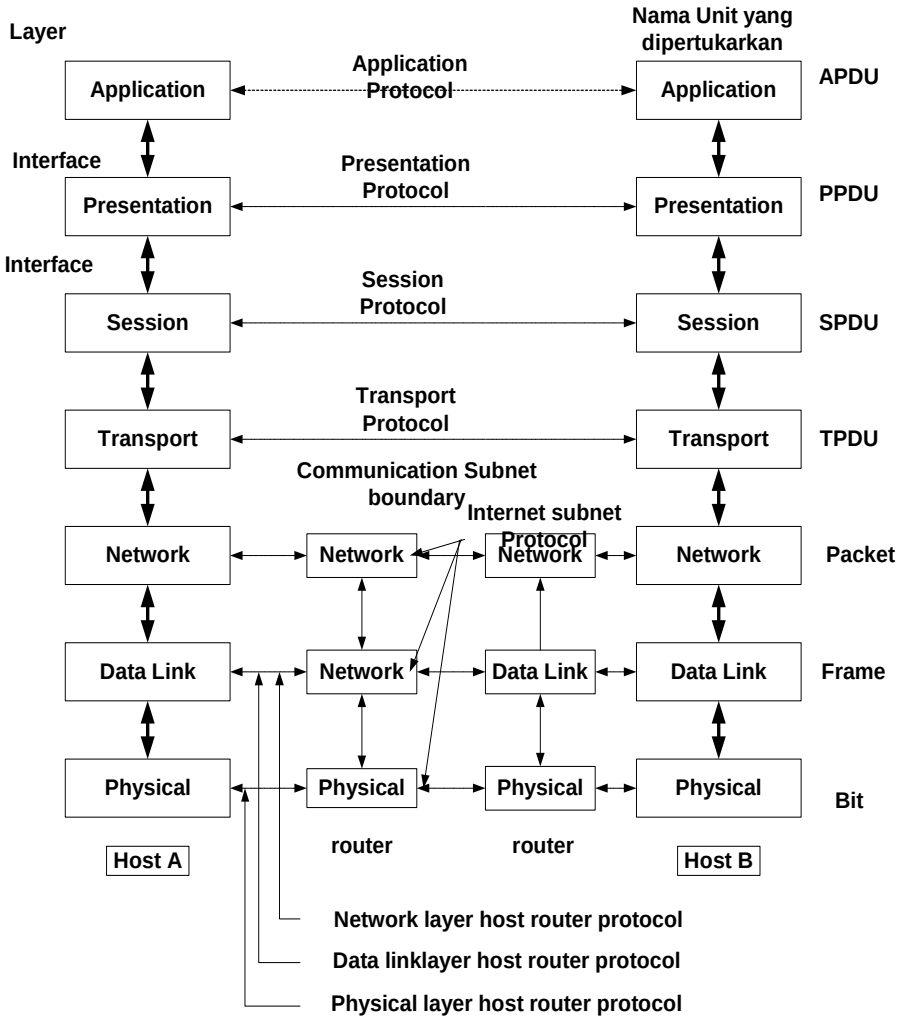
d. GAN (Global Area Network)

GAN merupakan suatau jarinagn yang menghubungkan Negara-negara diseluruh dunia. Kecepatan GAN bervariasi mulai dari 1,5 Mbps sampai dengan 100 Gbps dan cakupannya mencakupi ribuan kilometer.

2.3.3 Arsitektur Protokol

Open system interconnection (OSI), istilah ini mengacu pada standar untuk pertukaran informasi diantara sistem-sistem yang"terbuka" yaitu sistem yang menerapkan standar OSI. Fakta bahwa suatu sistem merupakan sistem yang terbuka tidak berpengaruh terhadap implementasi, teknologi, atau interkoneksi sistem, melainkan berhubungan dengan kesesuaian suatu sistem terhadap standar yang diterapkan. Berdasarkan hal ini, ISO (*organization for Standarization)*telah menspesifikasikan suatu model referensi OSI yang membagi-bagi fungsi komunikasi ke dalam tujuh lapis *(layer)*. Model OSI menyediakan secara konseptual kerangka kerja untuk komunikasi. Sebenarnya komunikasi dapat terjadi karena menggunakan protokol komunikasi. Dalam konteks jaringan (komunikasi data),sebuah protokol adalah suatu aturan formal dan kesepakatan yang menentukan bagaimana komputer bertukar informasi melalui sebuah media jaringan. Sebuah protokol mengimplementsikan salah satu atau lebih *layer-layer* OSI. Setiap layer ditugaskan untuk melaksankan suatu subset dari fungsi komunikasi tertentu yang pada gilirannya diterapkan di dalam suatu *data terminal equipment (DTE)* yang berkomunikasi dengan DTE yang lain. Setiap layer

mengandalkan layer yang ada dibawahnya untuk melaksanakan fungsi-fungsi yang lebih primitive serta sekaligus menyediakan layanan untuk mendukung layer yang berada diatasnya. Layer-layer ini dirancang agar setiap perubahan yang terjadi di dalam suatu layer tidak mempengaruhi layer yang lainnya. Model Referensi OSI diperlihatkan pada gambar 2.4



Gambar 2.4 Model Referensi OSI

a. *Physical Layer*

Lapisan ini bertanggung jawab untuk mengaktifkan dan mengatur Physical interface jaringan komputer. Pada lapisan ini, hubungan antar interface-interface dari perangkat keras diatur seperti hubungan antara DTE dan DCE. Interface yang didefinisikan pada lapisan ini antara lain 10BaseT, 100BaseTX,V35, X.21 dan *High Speed Serial Interface (HSSI).*

b. *Data Link Layer*

Lapisan ini mengatur topologi jaringan, *error notification* dan *flow control*. Tugas utama *data link layer* adalah sebagai fasilitas transmisi raw data dan mentransformasikan data tersebut ke saluran yang bebas dari kesalahan transmisi.

c. *Network Layer*

Network Layer berfungsi untuk mengendalikan operasi subnet dengan meneruskan paket-paket dari saru *node* ke *node* yang lain dalam jaringan. Masalah desain yang penting adalah bagaimana cara menetukan route pengiriman paket dari sumber ke tujuannya.

d. *Transport Layer*

Fungsi utama *Transport layer* adalah menerima data dari *session layer*, memecah data menjadi bagian-bagian yang lebih kecil bila perlu, meneruskan data ke *network layer,* dan manjamin bahwa semua potongan data tersebut bisa tiba disis lainnya dengan benar. Selain itu, semua hal tersebut harus dilaksnakan secara efisien, dan bertujuan dapat melindungi *layer-layer* bagian atas dari perubahan teknologi *hardware* yang tidak dapat dihindari.

e. *Session Layer*

Session Layer mengijinkan para pengguna untuk menetapkan *session* dengan pengguna lainnya. Layer ini membuka, mengatur dan menutup suatu *session* antara aplikasi-aplikasi.

f. *Presentation Layer*

Presentation Layer melakukan fungsi-fungsi tertentu yang diminta untuk menjamin penemuan sebuah penyelesaian umum bagi masalah tertentu. Selain memberikan sarana-sarana pelayanan untuk konversi, format dan enkripsi data, *Presentation Layer* juga beberja dengan file berformat ASCII, EBCDIC, JPEG, MPEG, TIFF, PICT, MIDI, dan Quick Time.

g. *Aplication Layer*

Lapisan ini memberikan saran pelayanan langsung ke *user*, yang berupa aplikasi-aplikasi dan mengadakan komunikasi dari program ke program. Jika kita mencari suatu *file* dari *file server* unruk digunakan sebagai aplikasi pengolah kata, maka proses ini bekerja melaui *layer* ini. Demikian pula ketika melakukan pengiriman *email*, *browse*, ke internet, *chatting*, membuka *telnet session*, atau menjalankan FTP, maka semua proses tersebut dilaksanakan di *layer* ini.

2.4 Algoritma Huffman

Algoritma Huffman, yang dibuat oleh seorang mahasiswa MIT bernama David Huffman pada tahun 1952, merupakan salah satu metode paling lama dan paling terkenal dalam kompresi teks. Algoritma Huffman menggunakan prinsip pengkodean yang mirip dengan kode Morse, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering

muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan.dengan rangkaian bit yang lebih panjang.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi data yang diinputkan) menjadi sekumpulan *codeword*, algoritma Huffman termasuk kedalam kelas algoritma yang menggunakan metode *static*. Metoda *static* adalah metoda yang selalu menggunakan peta kode yang sama, metoda ini membutuhkan dua fase (*two-pass*). Fase pertama untuk menghitung probabilitas kemunculan tiap simbol dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan di taransmisikan.

Sedangkan berdasarkan teknik pengkodean simbol yang digunakan, algoritma Huffman menggunakan metode *symbolwise*. Metoda *symbolwise* adalah metode yang menghitung peluang kemunculan dari setiap simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang jarang muncul.

2.4.1 Proses Pembentukan *Huffman Tree*

Kode Huffman pada dasarnya merupakan kode prefiks (*prefix code*). Kode prefiks adalah himpunan yang berisi sekumpulan kode biner, dimana pada kode prefik ini tidak ada kode biner yang menjadi awal bagi kode biner yang lain. Kode prefix biasanya direpresentasikan sebagai pohon biner yang diberikan nilai atau label. Untuk cabang kiri pada pohon biner diberi label 0, sedangkan pada cabang kanan pada pohon biner diberi label 1. Rangkaian bit yang terbentuk pada setiap lintasan dari akar ke daun merupakan kode prefiks untuk karakter yang berpadanan. Pohon biner ini biasa disebut *Huffman Tree*. *Huffman Tree* ini

dibentuk dengan cara menghitung terlebih dahulu frekuensi kemunculan karakter pada string. Kemudian mencari dua buah karakter yang memiliki nilai frekuensi terkecil dan gabungkan kedua karakter tersebut pada sebuah akar. Kemudian cari kembali karakter yang memiliki frekuensi terkecil, dan ulangi langkah tersebut sampai terbentuk sebuah pohon yang daun-daunnya berisi seluruh karakter pada string.

2.4.2 Proses Encoding

Encoding adalah cara menyusun *string* biner dari teks yang ada. Proses *encoding* untuk satu karakter dimulai dengan membuat *Huffman Tree* terlebih dahulu. Setelah itu, kode untuk satu karakter dibuat dengan menyusun nama *string* biner yang dibaca dari akar sampai ke daun *Huffman Tree*. Hasil dari penyusunan nama string biner tersebut menjadi kode biner yang baru untuk setiap karakter. Karakter yang memiliki frekuensi kemunculan yang besar akan memiliki kode biner yang lebih pendek dari pada karakter yang memiliki frekuensi kemunculan yang kecil.

2.4.3 Proses Decoding

Decoding merupakan kebalikan dari *encoding*. *Decoding* berarti menyusun kembali data dari *string* biner menjadi sebuah karakter kembali. *Decoding* dapat dilakukan dengan dua cara, yang pertama dengan menggunakan *Huffman Tree* dan yang kedua dengan menggunakan tabel kode Huffman. Melakukan proses decoding dengan menggunakan *Huffman Tree* dilakukan dengan cara menelusuri *Huffman Tree* dan mencatat string biner yang ada pada cabang sampai dengan

karakter yang dicari ditemukan. Sedangkan menggunakan tabel kode Huffman adalah dengan cara membuat sabuah table yang berisi karakter yang diencoding beserta string biner yang baru dibentuk dari *Huffman Tree.*

2.5 Algoritma LZ77

Algoritma LZ77 (Lempel Ziv 1977) merupakan algoritma kompresi yang bersifat *Loseless.* Algoritma ini dikembangkan oleh Abraham Lempel dan Jacob Ziv pada tahun 1977. Hal ini sesuai dengan nama yang diberikan pada algoritma kompresi tersebut yang merupakan singkatan dari nama penemunya. Algoritma LZ77 ini dalam melakukan kompresinya menggunakan teknik pengkamusan (*Dictionary Techniques)*. Algoritma LZ77 ini menggunakan dua buah buffer, yaitu *Dictionary Buffer* dan *Lookahead Buffer*. *Dictionary buffer* berfungsi untuk menyimpan symbol yang baru saja dikodekan dari inputan. Sedangkan *Lookahead Buffer* berfungsi untuk menyimpan symbol yang akan dikodekan. Hasil atau output dari proses encoding pada algoritma LZ77 ini terdiri dari tiga unsur atau tiga bit. Bit yang pertama berisi nilai dari posisi yang *match* atau cocok antara *dictionary buffer* dan *lookahead buffer*. Bit yang kedua berisi nilai dari panjang karakter yang *match* atau cocok antara *dictionary buffer* dan *lookahead buffer*. Bit yang ketiga berisi nilai dari karakter yang tidak *match* atau tidak cocok antara *dictionary buffer* dan *lookahead buffer.*

2.5.1 Proses Encoding

Proses encoding pada algoritma LZ77 ini dengan cara mencari nilai P,L dan S. Nilai P adalah posisi karakter yang sama pada *lookahead buffer* dengan *dictionary buffer*, nilai L adalah jumlah karakter yang sama pada *lookahead buffer* dengan *dictionary buffer* dan nilai S adalah karakter yang akan disimpan pada *dictionary buffer*. Output dari proses encoding pada algoritma LZ77 ini adalah berupa string gabungan dari nilai-nilai P,L dan S sampai dengan tidak ada lagi karakter pada *lookahead buffer*.

2.5.2 Proses Decoding

Proses decoding pada algoritma LZ77 ini dilakukan dengan cara mengambil 3 buah karakter dan kemudian digabungkan menjadi sebuah kelompok karakter. Ssetiap kelompok karakter yang telah dibuat berisikan nilai P, L dan S. Nilai P adalah karakter pertama pada kelompok karakter, nilai L adalah karakter kedua pada kelompok karakter dan nilai S adalah karakter ketiga pada kelompok karakter. String diambil dengan cara mencari posisi karakter pada *dictionary buffer* sesuai dengan nilai P, lalu mengambil banyaknya karakter yang akan diambil pada *dictionary buffer* sesuai dengan nilai S dan gabungkan karakter yang telah diambil dengan nilai S. Hasil dari proses decoding ini adalah string gabungan dari karakter-karakter yang telah diambil dari *dictionary buffer*.

2.6 Algoritma Deflate

Algoritma deflate merupakan algoritma persilangan antara algoritma Huffman dan algoritma LZ77. Dalam proses kompresinya, algoritma deflate ini terlebih dahulu melakukan proses pengelompokan karakter dengan menggunakan algoritma LZ77. Kemudian hasil dari pengelompokan karakter tersebut dikompresi lagi dengan menggunakan algoritma Huffman (*Huffman Tree)*. Algoritma deflate ini bersifat *loseless Compression*. Hal ini karena algoritma deflate ini menggabungkan dua algoritma kompresi yang bersifat *loseless.*

2.6.1 Proses Encoding

Proses encoding dari algoritma deflate ini dilakukan pada dua tahap. Tahap pertama adalah melakukan proses pembuatan blok-blok atau penyingkatan karakter dengan menggunakan algoritma LZ77. Tahap kedua adalah mengambil hasil penyingkatan karakter dari algoritma LZ77 dan melakukan proses kompresi dengan menggunakan *Huffman Tree* terhadap karakter tersebut. Hasil dari proses encoding algoritma deflate ini adalah berupa karakter yang telah disingkat yang merupakan hasil dari encoding algoritma LZ77 dan memiliki kode biner yang lebih pendek yang merupakan hasil dari encoding algoritma Huffman.

2.6.2 Proses Decoding

Proses decoding pada algoritma deflate merupakan kebalikan dari proses encodingnya. Langkah pertama adalah melakukan proses decoding dengan menggunakan algoritma Huffman. Kemudian langkah kedua adalah mengambil hasil dari proses decoding dengan menggunakan algoritma Huffman dan melakukan proses decoding kembali dengan menggunakan algoritma LZ77.

2.7 Flowchart

Flowchart adalah suatu alat yang cukup efesien untuk menggambarkan suatu algoritma agar dapat dengan mudah di implementasilakan pada suatu aplikasi. Tujuan utama penggunaan flowchart adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana. Metode ini menggunakan simbol-simbol tertentu yang mudah dimengerti dan telah distandarisasi dalam menyelesaikan masalah yang dihadapi.