BAB 2
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 bahan-bahan yang digunakan dalam melakukan penelitian ini.

2.1 Data

Data adalah representasi dari fakta, konsep, atau instruksi dalam format tertentu yang digunakan untuk komunikasi, intepretasi, atau diproses oleh manusia[5]. Data juga dapat diartikan suatu kumpulan angka, karakter, gambar yang sebelumnya tidak memiliki arti apa-apa hingga diproses menjadi informasi melalui pemrosesan data[6].

2.2 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[7]. Contoh kompresi sederhana yang biasa kita lakukan dalam kehidupan sehari-hari adalah misalnya kita menyingkatkan kata-kata yang sering kita gunakan tapi kata tersebut telah dipahami oleh setiap orang atau biasa digunakan secara umum. Misalnya : kata “yang” dikompres atau disingkat menjadi kata “yg”.

Secara mendasar standar-standar teknik kompresi data memiliki 2 teknik kompresi yaitu teknik kompresi *lossy* dan teknik kompresi *loseless*. Untuk teknik kompresi secara *lossy* ada sebagian informasi yang dihilangkan di dalam data tersebut pada saat proses kompresi terjadi. Sedangkan untuk teknik kompresi secara *loseless* kebalikan dari teknik kompresi secara *lossy* yaitu tidak adanya informasi yang hilang pada saat proses kompresi terjadi.

2.2.1 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 terjadinya proses kompresi).
c. *Ratio Compress* (ukuran data setelah dilakukannya proses kompresi) Berikut ini adalah persamaan menentukan nilai rasio kompresi[8] :

$$\textit{Ratio Compress} = \frac{ukuran\ data\ setelah\ dikompresi}{ukuran\ data\ sebelum\ dikompresi}\ x\ 100\ \% \qquad (2,1)$$

d. *Optimaly* (perbandingan apakah ukuran *file* setelah dilakukannya proses kompresi sama atau tidak sama dengan *file* sebelum dilakukannya proses kompresi).

2.2.2 Metode Kompresi Data

Proses kompresi memiliki beberapa jenis metode dalam melakukan proses kompresi tersebut. Proses kompresi tersebut dibedakan berdasarkan tipe peta kode kompress data yang digunakan untuk mengubah pesan awal(isi *file input)* menjadi sekumpulan *code word* dan berdasarkan teknik pengkodean/pengubahan symbol yang digunakan.

Pada proses kompresi yang berdasarkan tipe peta kode kompress data yang digunakan untuk mengubah pesan awal(isi *file input)* menjadi sekumpulan *code word*, metode proses kompresi ini terbagi menjadi dua kelompok, yaitu :

a. Metode statik : menggunakan peta kode yang selalu sama. Metode ini membutuhkan dua fase(*two-pass*). Dimana fase pertama untuk menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya dan pada 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 *one-pass*. Karena isi *file* selama dikompres hanya diperlukan satu kali pembacaan terhadap isi *file*. Contoh : algoritma *lemple ziv welch* (LZW) dan *Dinamic Markov Compression* (DMC).

Sedangkan pada proses kompresi 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 sering muncul contoh : algoritma *Huffman*.
b. metode *dictionary* : metode yang menggantikan karakter/fragmen dalam *file input* dengan indeks lokasi dari karakter/fragmen tersebut dalam sebuah kamus (*dictionary*), contoh : algoritma LZSS.
c. metode *predictive* : metode yang menggunakan model *finite-context* atau *finite-state* untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya, contoh : algoritma DMC.

2.2.3 Tipe Kompresi Data

Pada proses kompresi data, secara umum proses kompresi ini memiliki dua tipe, yaitu :

1. *Lossy*

Lossy Compression adalah tipe kompresi data yang memiliki sifat data sumber (asli) dan data setelah dikompresi apabila dilakukannya proses dekompresi kembali hasilnya tidak sesuai dengan data sumbernya. Umumnya tipe kompresi data ini digunakan untuk proses kompresi pada data bertipe *image*[9].

Keuntungan pada tipe kompresi ini adalah membuat nilai rasio kompresi (perbandingan antara ukuran *file* asli dengan *file* hasil proses

kompresi) cukup tinggi. Hal ini dikarenakan cara kerja *lossy* adalah dengan mengeliminasikan beberapa data dari suatu *file*, sehingga menyebabkan data pada suatu *file* yang telah melewati proses kompresi hilang ketika proses dekompresi. Namun data yang dieliminasikan biasanya adalah data yang kurang diperhatikan atau di luar jangkauan manusia, sehingga pengeliminasian data tersebut kemungkinan besar tidak akan mempengaruhi proses interaksi manusia 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 masih bisa mentoleransi tipe kompresi ini adalah citra, audio dan video.

2. Loseless

Loseless Compression adalah tipe kompresi data yang memiliki sifat data sumber (asli) dan data setelah dikompresi lalu dilakukannya proses dekompresi kembali akan memiliki struktur *file* yang sama[9].

2.3 Lemple Ziv Storer Symanski (LZSS)

Algoritma *Lemple Ziv Storer Symanski* (LZSS) adalah algoritma kompresi data yang bertipe *loseless* yang dimodifikasi dari algoritma LZ77, dinamai penciptanya James Storer dan Thomas Symanski (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 algoritma *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[7].

2.3.1 **Algoritma Lemple Ziv Storer Symanski (LZSS)**

Cara kerja algoritma *Lemple Ziv Storer Symaski* (LZSS) terdiri dari sebuah *literal bytes* atau pasangan (*offset,match length*). *Literal bytes* merupakan *input byte* sederhana yang dikopikan secara langsung ke *output*. pasangan (*offset,match length*) menjelaskan proses kembali ke jarak *offset* suatu *input* data yang nantinya akan ditemukan sebuah *match length* dari data tersebut, algoritma ini menggunakan dua buah *flag* pada *output*, yaitu :

1. Jika *flag* "0", *byte* berikutnya adalah *literal*.
2. Jika *flag* "1", merupakan pasangan (*offset,match length*).

Sebagai Contohnya :

1. *Input* data = "ABCDABCA"
   Mulai dengan byte "A", apakah telah ada sebelumnya? "TIDAK". Enkodenya adalah sebuah *literal byte* "0" diikuti data "A". *Output* Data = 0"A". Proses terus berlangsung sampai byte berikutnya sama dengan byte sebelumnya. Dengan kata lain *output* data menjadi : 0 "A " 0 " B " 0" C " 0 " D".
2. *Input* data = "ABCDABCA"
   Byte berikutnya adalah "A", apakah telah ada sebelumnya ? "YA" 4 *bytes* sebelumnya (*offset* = 4). Ada berapa bytes yang sama? 3 *byte* yang sama "ABC", maka akan dikodekan "1" (untuk mengidentifikasi bahwa ada data yang sama) diikuti (4,3). *Output* Data= 0 "A" 0 "B" 0 "C" 0 "D" 1 (4,3).
3. *Input* data = "ABCDABCA"
   Byte berikutnya adalah "A", apakah telah ada sebelumnya ? "YA" 3 *bytes* sebelumnya (*offset* = 3). Ada berapa *bytes* yang sama? 1 *byte* yang sama "A", maka akan dikodekan "1" diikuti (3,1). *Output* Data = 0 "A" 0 "B" 0 "C" 0 "D" 1 (4,3) 1 (3,1).

Di bawah ini adalah tabel untuk *Input Stream* yang disajikan pada tabel 2.1.

Tabel 2.1 *Input Stream*

Pos	1	2	3	4	5	6	7	8	9	10	11
Char	A	A	B	B	C	B	B	A	A	B	C

Berikut ini adalah keterangan dari tabel 2.1 :

Input stream : barisan karakter yang akan dikompres

Pos : posisi dari setiap karakter pada *input stream*

Char : deretan karakter dalam *input stream*

Di bawah ini adalah tabel untuk proses *Encoding* yang disajikan pada tabel 2.2.

Tabel 2.2 Prosess *Encoding*

Step	Pos	Match	Output
1	1	--	A
2	2	A	A
3	3	--	B
4	4	B	B
5	4	--	C
6	6	BB	(3,2)
7	8	AAB	(7,3)
8	11	C	C

Berikut ini adalah keterangan dari tabel 2.2 :

Proses *Encoding* : proses pengkodean input *Stream*

Pos : posisi dari setiap karakter pada *input stream*

Step : langkah proses pengkodean;

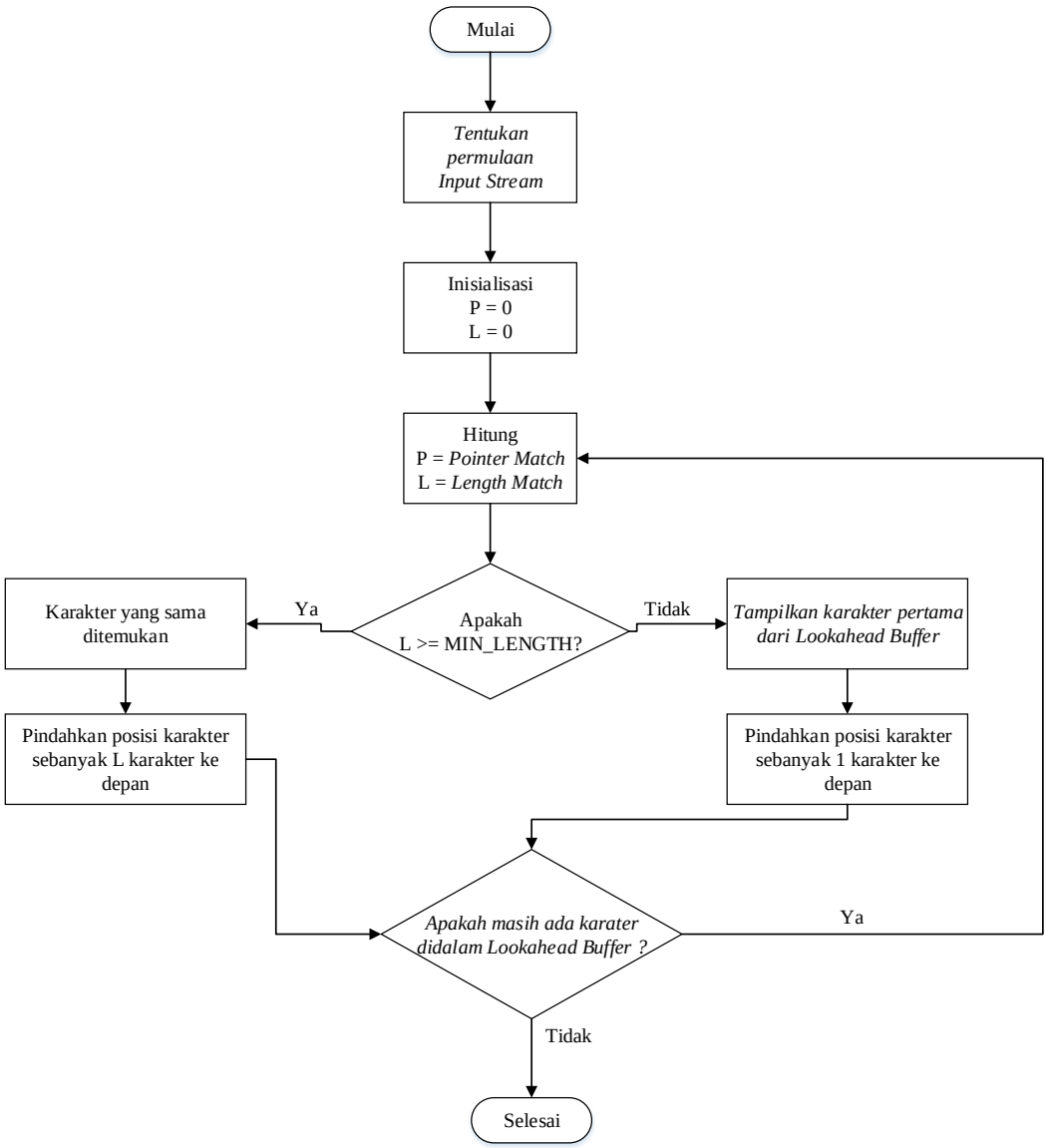
Match *:* karakter yang sama yang ditemukan

Output : barisan *simbol* hasil proses e*ncoding*

Algoritma untuk proses *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 : keluarannya P dan bergerak ke posisi L karakter.
    b. Jika TIDAK : keluarannya karakter pertama pada *lookahead buffer* dan bergerak satu posisi karakter ke depan.
4. Jika karakter *input stream* banyak, ulangi langkah 2

Dari uraian di atas, maka diagram alir untuk proses *encode* dengan menggunakan algoritma LZSS dapat disajikan sebagai berikut :



Gambar 2. 1 Diagram Alir Proses LZSS

2.3.2 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 *byte* maksimum yang dapat dicocokan.

Untuk aturan pertama hanya membutuhkan dua *flag* yaitu *literal* atau *match*, jadi hanya menggunakan bit tunggal. Jika menggunakan bit

tunggal maka seluruh *byte* tidak selalu ditulis tapi hanya sebagian saja. Sebelum proses kompresi dilakukan, dibutuhkan suatu fungsi kode untuk membaca dan menulis jumlah variabel bit dari atau ke data *stream.*

Sebagai contoh : batas *offset* menggunakan 4 bits, jadi panjangnya antara 0-31, sedangkan untuk len menggunakan 3 bits jadi panjangnya antara 0-7. Keluarannya 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.*

2.3.3 Cyclic Redudancy Checking (CRC)

Cyclic Redudancy Checking (CRC) merupakan kaidah untuk meralat data yang akan dikirimkan melalui saluran komunikasi, pengirim mengaplikasikan *polynominal* 16 atau 32 bit pada blok data yang akan dikirim dan menambahkan hasil *polynominal* yang sama pada data dan membandingkan hasilnya dengan yang diterimanya. Jika keduanya sama maka data telah berhasil dikirim. Jika tidak, pengirim harus mengirimkan kembali data semula yang diralat.

Consultative Committee for International Telephony and Telegraphy (CCITT) mempunyai kemampuan *polynominal* 16 bit yang digunakan protokol CRC. IBM SDLC menggunakan protokol CRC-16, CRC akan menghasilkan ralat 1 atau 2 bit dan keberhasilannya adalah 99,998%[10].

2.4 Algoritma Lemple Ziv Markov Chain (LZMA)

LZMA adalah pembaharuan dari algoritma LZ77, dengan menggunakan *markov chain* dan *entropy coding*[7]. Metode algoritma ini menggunakan bagian *stream input* sebagai kamus yang akan digunakan dalam kompresi data. *Encoder* menerima *input* dari kanan ke kiri bersamaan dengan keseluruhan data yang akan melalui proses kompresi. Metode ini juga disebut dengan istilah *sliding windows*. Pada *windows* ini terbagi menjadi dua bagian, bagian sebelah kiri dinamakan *search buffer* dan

sebelah kanan dinamakan *look-ahead buffer*, yang berisi teks yang akan dikompresi. Seperti halnya LZ77, LZMA juga menghasilkan suatu keluaran yang menyimpan *distance*, *length*, dan *next symbol* sebagai suatu kesatuan berkas hasil kompresi. *Distance* merupakan jarak antara karakter yang dicari pada *look-ahead buffer* yang sama dengan karakter pada *search buffer*. *Length* adalah panjang dari rangkaian karakter terpanjang pada *search buffer* yang cocok pada *look-ahead buffer*. *Next symbol* adalah satu karakter pada *look-ahead buffer* setelah rangkaian karakter terpanjang terpilih.

Prinsip kompresi pada LZMA sama dengan teknik kompresi pada deflate, tetapi menggunakan proses *range coding* sebagai pengganti *Huffman coding*. Pada kasus nyata, hasil dari proses kompresi menggunakan algoritma LZMA berupa suatu *file* yang berisi rangkaian karakter representasi dari satu kesatuan berkas hasil kompresi yang terdiri dari *distance*, *length*, dan *next symbol*.

2.4.1 Algoritma Lemple Ziv Markov Chain (LZMA)

Cara kerja algoritma *Lemple Ziv Markov Chain* (LZMA) pada proses kompresi adalah dimulai dengan memilih *file* yang akan dikompresi. *File* yang dipilih akan dikenali sebagai *input stream* yaitu deretan simbol/karakter yang ukuran panjangnya sesuai dengan besar *file*. Algoritma ini akan membaca *input stream* dengan menggunakan *search buffer* dengan ukuran 2^w bit dan *lookahead buffer*. Dimana w adalah nilai pangkat yang dibatasi oleh jumlah *memory* dalam perangkat keras.

Proses kompresi diawali dengan membaca *input stream* dalam satuan *bytes* dan mencatat n *bytes* terakhir yang dibaca, jika menemukan susunan kata yang sama maka akan memberikan *output* nilai yang disebut dengan *token*. Dimana *token* memiliki struktur sebagai berikut :

T = (x,y,Z)

x = *offset* (2 bit)

y = *length* (2 bit)

z = *next symbol* (1 *byte*)

Selama proses *encode* ini, program juga merekam probabilitas setiap karakter yang bersebelahan serta membentuk *markov model* dalam susunan *directed graph*.

Sebagai contoh, data yang akan dikompresi sebagai berikut : AABBCBBAABC. Di bawah ini adalah tabel untuk *Input Stream* yang disajikan pada tabel 2.3.

Tabel 2.3 Input Stream

Pos	1	2	3	4	5	6	7	8	9	10	11
Char	A	A	B	B	C	B	B	A	A	B	C

Berikut ini adalah keterangan dari tabel 2.3 :

Input stream : barisan karakter yang akan dikompres

Pos : posisi dari setiap karakter pada *input stream*

Char *:* deretan karakter dalam *input stream*

Di bawah ini adalah tabel untuk proses *Encoding* yang disajikan pada tabel 2.4.

Tabel 2.4 Proses Encoding Algoritma LZMA

Step	Pos	Match	Token
1	1	--	(0,0,”A”)
2	3	A	(1,1,”B”)
3	5	B	(1,1,”C”)
4	6	BBB	(3,2,”A”)
5	11	ABA	(7,2,”C”)

Berikut ini adalah keterangan dari tabel 2.4 :

Proses *Encoding* : proses pengkodean input *Stream*

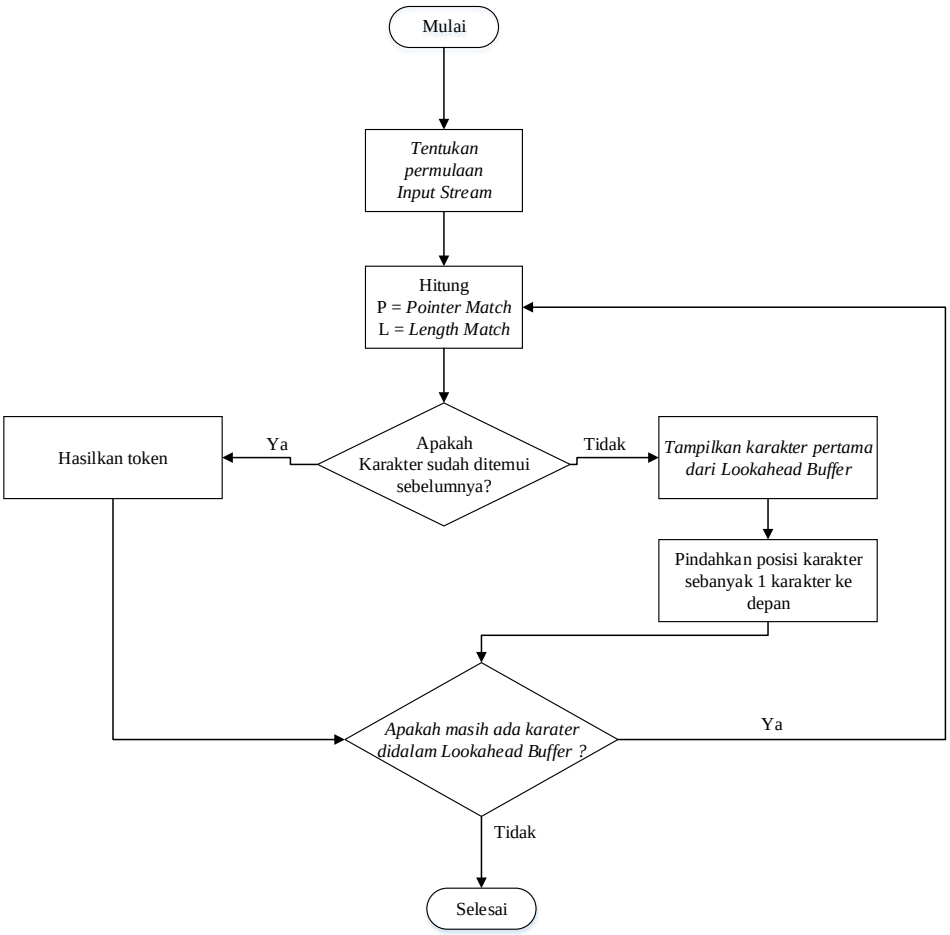
Pos : posisi dari setiap karakter pada *input stream*

Step : langkah proses pengkodean;

Match : *karakter yang sama yang ditemukan*

Token : barisan *simbol* hasil proses e*ncoding* yang terdiri dari nilai *offset, length* dan karakter selanjutnya

Dari uraian di atas, maka diagram alir untuk proses *encode* dengan menggunakan algoritma LZMA dapat disajikan sebagai berikut :



Gambar 2. 2 Diagram Alir Proses LZMA

2.4.2 Entropy Coding

Entropy coding adalah penugasan kode-kode ke dalam simbol sehingga memiliki panjang kode yang sama dengan probabilitas dari jumlah simbolnya.

2.5 Komunikasi Data

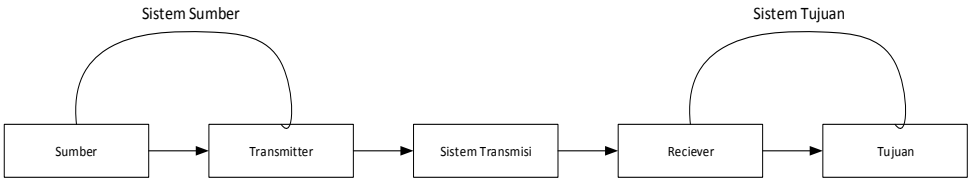
Komunikasi data adalah proses pengiriman informasi diantara dua titik menggunakan kode biner melewati saluran transmisi dan peralatan *switching* dapat terjadi antara komputer dengan komputer, komputer dengan terminal atau komputer dengan peralatan lainnya[11]. Komunikasi data merupakan gabungan dari teknik telekomunikasi dengan teknik pengolahan data.

Adapun tujuan dari komunikasi data adalah sebagai berikut :

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

2.5.1 **Model Komunikasi Data**

Komunikasi data berkaitan dengan pertukaran data diantara dua perangkat yang terhubung secara langsung yang memungkinkan adanya pertukaran data antar kedua belah pihak. Secara keseluruhan proses komunikasi data dapat dilihat pada Gambar 2.1



Gambar 2. 3 Komunikasi Data

Pada gambar di atas terdapat elemen-elemen dalam kunci model tersebut :

1. *Source* (sumber) : Alat ini membangkitkan data sehingga dapat ditransmisikan, contoh telepon, komputer.

2. *Transmitter* (pengirim) : Biasanya data yang dibangkitkan dari sumber tidak ditransmisikan secara 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 network*) yang menghubungkan antara sumber dengan tujuan (*destination*).
4. *Reciever* : merubah sebuah sinyal yang didapat menjadikan sebuah data sehingga dapat diteruskan kepada tujuan.
5. Tujuan : menangkap data yang dihasilkan oleh *receiver*.

2.6 Bandwidth

Definisi dari *Bandwidth* adalah banyaknya ukuran suatu data atau informasi yang dapat mengalir dari suatu tempat ke tempat lain dalam sebuah *network* di waktu tertentu. *Bandwidth* dapat dipakai untuk mengukur baik aliran data analog maupun data digital. Sekarang sudah menjadi umum jika kata bandwith lebih banyak dipakai untuk mengukur aliran data digital.

Satuan yang dipakai untuk bandwidth adalah Bits Per Second atau sering disingkat Bps. Seperti diketahui bahwa bit atau binary digit adalah basis angka yang terdiri dari 0 dan 1. Satuan ini menggambarkan berapa banyak bit (angka 0 dan 1) yang dapat mengalir dari satu tempat ke tempat yang lainnya dalam setiap detiknya melalui suatu media[11].

2.7 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[11]. 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.

2.7.1 Jenis-jenis jaringan komputer

Terdapat beberapa jenis jaringan komputer, pada pembagian jenis jaringan komputer ini terbagi berdasarkan skala dan fungsinya. Berikut ini adalah jenis-jenis jaringan komputer :

1. Berdasarkan Skala

Secara umum jenis jaringan komputer berasarkan skalanya terbagi atas 3 jenis, yaitu :

a. *Local Area Network* (LAN)

Jaringan ini digunakan untuk menghubungkan komputer-komputer lokal, seluruh komputer yang terhubung ke jaringan terhubung pada satu pusat yang disebut *gateway*. LAN biasa ditemui pada jaringan-jaringan kecil dalam suatu ruangan atau lembaga tertentu. Dalam pengembangannya, LAN berkembang menjadi sebuah *Metropolitan Area Network* (MAN), yang sudah melibatkan lebih dari satu *gateway*, dan biasanya telah memiliki sebuah *server* utama. MAN biasanya diterapkan pada Sistem Informasi perkotaan.

b. *Metropolitan Area Network* (MAN)

Metropolitan Area Network (MAN), pada dasarnya merupakan versi LAN yang berukuran lebih besar dan biasanya menggunakan teknologi yang sama dengan LAN. MAN dapat mencakup kantor-kantor perusahaan yang letaknya berdekatan atau juga sebuah kota dan dapat dimanfaatkan

untuk keperluan pribadi (swasta) atau umum. MAN mampu menunjang data dan suara, bahkan dapat berhubungan dengan jaringan televisi kabel.

c. *Wide Area Network* (WAN)

Jaringan komputer skala luas (WAN) merupakan pengembangan dari MAN dan telah melibatkan lebih dari satu *server* utama, masing masing *server* utama saling terhubung dan setiap komputer yang terhubung ke jaringan akan dapat saling mengakses *server* tersebut. WAN biasanya mencakup sebuah area yang sangat luas, bahkan antarnegara. Dalam perkembangannya, WAN akan berkembang menjadi sebuah jaringan global yang biasa dikenal sebagai *Interconnected-Networking* (Internet). Untuk menghubungkan jaringan lokal dengan jaringan lokal lain maupun dengan jaringan berskala luas (WAN), diperlukan sebuah perangkat khusus untuk mengatur *gateway*-nya, yaitu *Router*.

2. Berdasarkan Fungsi

Secara umum jenis jaringan komputer berasarkan fungsinya terbagi atas 2 jenis, yaitu :

a. Jaringan *peer to peer* (P2P) atau point-*to-point*

Kedudukan setiap komputer yang terhubung dalam jaringan adalah sama. Tidak ada komputer yang menjadi pelayan utama (*server*). Sehingga semua komputer dalam jaringan dapat saling berkomunikasi dan berbagi penggunaan perangkat keras dan perangkat lunak .

b. Jaringan *client-server*

Pada jaringan *client-server* terdapat sebuah komputer yang mengatur semua fasilitas yang tersedia dalam jaringan komputer, seperti komunikasi, penggunaan bersama perangkat keras dan perangkat lunak serta mengontrol jaringan. Komputer ini dinamakan *server*. Semua komputer lain selain *server* disebut *client*.

2.7.2 Topologi jaringan komputer

Topologi merupakan diagram yang mewakili cara komputer terhubung dalam jaringan[11]. Terdapat bermacam-macam topologi di

dalam teori jaringan komputer, antara lain beberapa macam topologi yang sering digunakan adalah topologi *star*, BUS, *ring* dan *tree*.

2.8 Object Oriented Analysis And Design (OOAD)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. Pengertian berorientasi objek berarti bahwa cara mengorganisasikan perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data perilakunya.

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu:

1. Analisis berorientasi objek (OOA)

   Metode analisis yang memerika *requirement* (syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup perusahaan. *Unified Modeling Language* (UML) adalah bahasa untuk mengspesifikasi, memvisualisasi, membangun dan mendokumentasi *artefacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak), seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya. UML merupakan bahasa standar untuk penulisan *blueprint software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak.

   UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat.

   a. Use Case Diagram

      Menggambarkan sejumlah eksternal aktor dan hubungannya ke *use case* yang diberikan oleh sistem.

b. Activity Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainya seperti *use case*.

c. Class Diagram

Menggambarkan struktur dan deskripsi *class, package* (paket) dan objek beserta hubungan satu sama lain seperti *containment* (penahanan), pewarisan, asosiasi dan lain-lain

d. Sequence Diagram

Menggambarkan kolaborasi dinamis antara sejumlah objek dan untuk menunjukan rangkaian pesan yang dikirim antara objek juga interaksi. Sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem [12].

2. Desain berorientasi objek (OOD)

Metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

2.9 Teknologi .NET

Teknologi yang dikembangkan oleh Microsoft yang mempunyai arti memberikan pemakai akses ke informasi, *file*, program, setiap tempat, setiap saat, setiap *platform* dan setia perangkat. Pemakai tidak perlu tahu di mana informasi berada atau bagaimana cara memanggilnya[13].

2.10 C#

Bahasa Pemrograman berbasis objek terbaru dengan menggunakan teknologi .NET untuk membuat aplikasi berbasis *desktop*, aplikasi berbasis web, layanan web XML dan aplikasi berbasis *mobile*. C# dibangun di atas fondasi .NET *framework* serta dibangun berdasarkan metode yang digunakan bahasa pemrograman C++ dan C[14].