BAB II

LANDASAN TEORI

2.1 Konsep Dasar Jaringan

Jaringan komputer adalah sekelompok komputer yang saling dihubungkan sehingga antara satu komputer dengan komputer yang lain dapat berbagi data atau berbagi sumber daya *(sharing Resource)* [2].

Komputer - komputer yang terhubung dalam jaringan computer harus memakai aturan komunikasi (protokol) yang sama. Hal ini dimaksudkan agar masing – masing komputer dapat berkomunikasi yang baik dengan komputer yang lainnya. Protocol yang menjadi standar International adalah *ITCP/IP (Transmission Control Protocol / Internet Protocol).*

2.1.1 Berdasarkan Skala

Secara umum jaringan komputer 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*.

2.1.2 Berdasarkan Fungsi

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 .

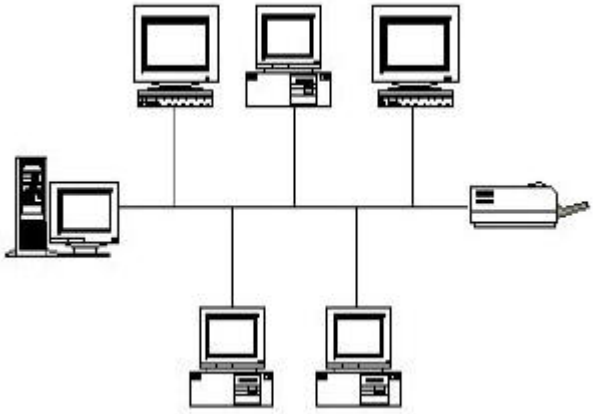
b. *Jaringan client-server*

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.

2.1.3 Topologi Jaringan Komputer

Topologi merupakan diagram yang mewakili cara komputer terhubung dalam jaringan[3]. Terdapat bermacam-macam topologi di dalam teori jaringan komputer, namun dalam laporan ini hanya beberapa topologi yang dibahas. Karena topologi ini masih sering digunakan oleh perusahaan ataupun pengguna jaringan komputer lainnya:

1. *Topologi Bus*



Gambar 2.1 Topologi Bus

Pada topologi bus seluruh komputer dalam jaringan terhubung dalam sebuah bus atau jalur komunikasi data utama / *backbone* (berupa kabel) seperti yang telihat pada gambar 2.1. Komputer dalam jaringan berkomunikasi dengan cara mengirim dan mengambil data melalui bus.

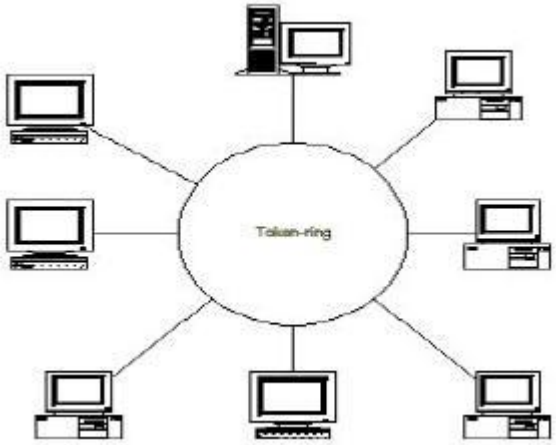
Keunggulan :

a. Hemat kabel.
b. Layout kabel sederhana.
c. Mudah dikembangkan.

Kelemahan :

a. Deteksi dan isolasi kesalahan sangat kecil.
b. Kepadatan lalu lintas data.
c. Bila salah satu *client* rusak, maka jaringan tidak bisa berfungsi.
d. Diperlukan *repeater* untuk jarak jauh.

2. *Topologi Ring*



Gambar 2.2 Topologi Ring

Sesuai dengan namanya, *ring* atau cincin, seluruh komputer dalam jaringan terhubung pada sebuah jalur data yang menghubungkan komputer satu dengan lainnya secara sambung-menyambung sedemikian rupa sehingga menyerupai sebuah cincin seperti yang terlihat pada gambar 2.2. Topologi ini mirip dengan hubungan seri pada rangkaian listrik, dengan kedua ujung dihubungkan kembali, sehingga jika salah satu komputer mengalami gangguan, maka hal itu akan mempengaruhi keseluruhan jaringan. Dalam sistem jaringan ini, data dikirim secara berkeliling sepanjang jaringan (*ring*). Setiap komputer yang ingin mengirimkan data ke komputer lain harus melalui ring ini.

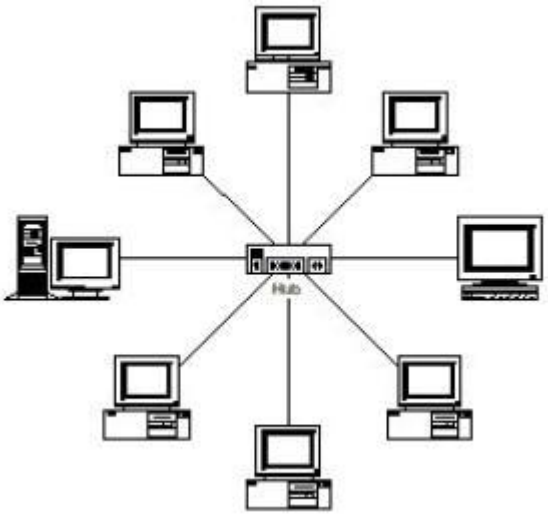
Keunggulan :

a. Hemat kabel.

Kelemahan :

a. Peka terhadap kesalahan.
b. Pengembangan jaringan lebih kaku.

3. *Topologi Star*



Gambar 2.3 Topologi Star

Dalam topologi ini masing-masing komputer dalam jaringan dihubungkan ke sebuah konsentrator dengan menggunakan jalur yang berbeda-beda, sehingga jika salah satu komputer mengalami gangguan, jaringan tidak akan terpengaruh (Gambar 2.3). Komunikasi di dalam jaringan diatur oleh konsentrator, berupa *hub* maupun *switch*.

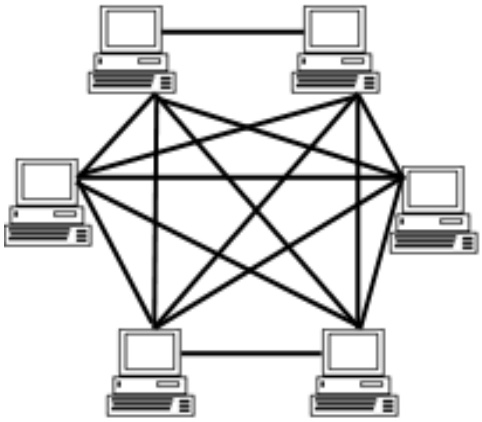
Keunggulan :

a. Paling fleksibel.
b. Pemasangan/perubahan stasiun mudah dan tidak mengganggu bagian jaringan lain.
c. Kontrol terpusat.
d. Kemudahan deteksi dan isolasi kesalahan/kerusakan.
e. Kemudahan pengelolaan jaringan.

Kelemahan :

a. Boros kabel.
b. Perlu penanganan khusus.
c. Kontrol terpusat (HUB) jadi elemen kritis.

4. *Topologi Mesh*



Gambar 2.4 Topologi Mesh

Topologi ini sering disebut “pure peer-to-peer”, sebab merupakan implementasi suatu jaringan komputer yang menghubungkan seluruh komputer secara langsung seperti yang terlihat pada gambar 2.4. Saat ini sangat jarang digunakan sebab rumit dan tidak praktis.

Keunggulan :

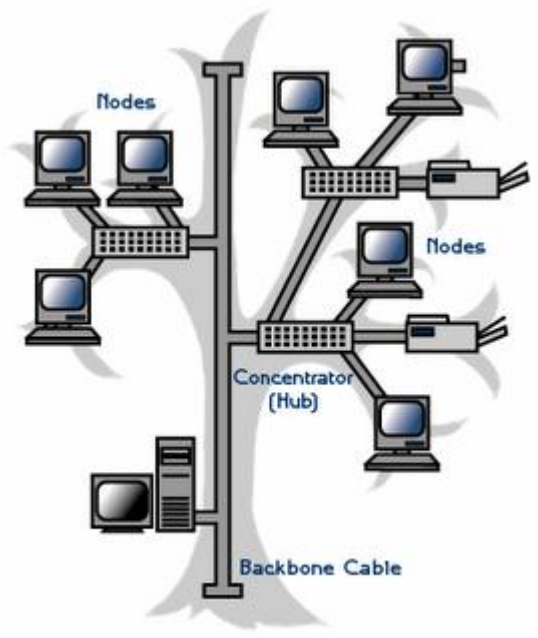
a. Hubungan dedicated links menjamin data langsung dikirimkan ke komputer tujuan tanpa harus melalui komputer lainnya sehingga dapat lebih cepat karena satu link digunakan khusus untuk berkomunikasi dengan komputer yang dituju saja (tidak digunakan secara beramai-ramai/sharing).

b. Memiliki sifat Robust, yaitu Apabila terjadi gangguan pada koneksi komputer A dengan komputer B karena rusaknya kabel koneksi (links) antara A dan B, maka gangguan tersebut tidak akan mempengaruhi koneksi komputer A dengan komputer lainnya.
c. Privacy dan security pada topologi mesh lebih terjamin, karena komunikasi yang terjadi antara dua komputer tidak akan dapat diakses oleh komputer lainnya.
d. Memudahkan proses identifikasi permasalahan pada saat terjadi kerusakan koneksi antar komputer.

Kelemahan :

a. Membutuhkan banyak kabel dan Port I/O. semakin banyak komputer di dalam topologi mesh maka diperlukan semakin banyak kabel links dan port I/O (lihat rumus penghitungan kebutuhan kabel dan Port).
b. Karena setiap komputer harus terkoneksi secara langsung dengan komputer lainnya maka instalasi dan konfigurasi menjadi lebih sulit.
c. Banyaknya kabel yang digunakan juga mengisyaratkan perlunya space yang memungkinkan di dalam ruangan tempat komputer-komputer tersebut berada.

5. *Topologi Tree*



Gambar 2.5 Topologi Tree

Topologi jaringan ini disebut juga sebagai topologi jaringan bertingkat. Topologi ini biasanya digunakan untuk interkoneksi antar sentral denganhirarki yang berbeda. Untuk hirarki yang lebih rendah digambarkan pada lokasi yang rendah dan semakin keatas mempunyai hirarki semakin tinggi (Gambar 2.5). Topologi jaringan jenis ini cocok digunakan pada sistem jaringan komputer .

Pada jaringan pohon, terdapat beberapa tingkatan simpul (node). Pusat atau simpul yang lebih tinggi tingkatannya, dapat mengatur simpul lain yang lebih rendah tingkatannya. Data yang dikirim perlu melalui simpul pusat terlebih dahulu. Misalnya untuk bergerak dari komputer dengan node-3 kekomputer node-7 seperti halnya pada gambar, data yang ada harus melewati node-3, 5 dan node-6 sebelum berakhir pada node-7.

Keungguluan jaringan model pohon seperti ini adalah, dapat terbentuknya suatu kelompok yang dibutuhkan pada setiap saat.

Kelemahannya adalah, apabila simpul yang lebih tinggi kemudian tidak berfungsi, maka kelompok lainnya yang berada dibawahnya akhirnya juga menjadi tidak efektif. Cara kerja jaringan pohon ini relatif menjadi lambat.

2.1.4 Pemanfaatan Jaringan Komputer

Pembentukan sebuah jaringan komputer sangan erat dengan manfaat yang dapat diperoleh dengan adanya jaringan tersebut[2].

1. **Bagi pakai (sharing) peralatan (*resources*)**

Dengan adanya jaringan komputer, maka pemakain beberapa peralatan komputer seperti *printer*, *hard disk*, *disket*, *scanner*, *CD-ROM* dan lainnya dapat dilakukan bersama-sama saling bergantian tanpa harus memindahkan posisi peralatan yang terpasang tersebut.

2. **Bagi pakai *software***

Hampir dalam setiap organisasi, kemampuan dalam melakukan bagi pakai berkas atau *file* data diperlukan setiap hari. Beberapa tipe *software* PC, khususnya program manajemen basis data atau *database*, didesain disamping agar bisa dipakai oleh satu pemakai, juga dimungkinkan untuk dipakai bersama-sama dengan pemakai lain dalam waktu yang bersamaan. Atau dengan kata lain, untuk mengakses dan meng-*update file-file* tadi. Paket yang lain, seperti program pengolah data (*word processor*) dan *spreadsheet*, kebanyakan didesaian hanya untuk satu pemakai yang dapat meng-*update* file.

3. Komunikasi

Komunikasi antar pemakai dalam suatu jaringan dapat dilakukan dengan menggunakan *e-mail* atau tele *conference*. Sehingga kebutuhan akan komunikasi antar pemakai dapat dipenuhi tanpa harus pindah dari tempat kerjanya. Selain itu pemakai e-mail dapat menekan pemakaian pulsa telepon.

4. Pemrosesan terpusat (terdistribusi)

Didalam suatu jaringan komputer, data dapat diolah secara terpusat atau secara terdistribusi. Pemrosesan secara terpusat dilakukan apabila sebuah data yang dibuat oleh tiap pemakai jaringan dikehendaki untuk disatukan dalam komputer pusat. Sebaliknya, pemrosesan terdistribusi dilakukan apabila suatu pekerjaan pengolahan data dari komputer pusat dapat dikerjakan oleh tiap pemakai berdasarkan spesialisasi bidang kerjanya.

5. Keamanan data

Keamanan data dapat diatur oleh *supervisor (administrator)* dengan pemberian hak akses, pembatasan waktu akses dan pemberian *password* untuk melindungi pemakaian komputer pusat.

2.2 Kompresi

Kompresi data adalah sebuah cara untuk memadatkan data sehingga hanya memerlukan ruangan penyimpanan lebih kecil sehingga lebih efisien dalam menyimpannya atau mempersingkat waktu pertukaran data tersebut[4].

2.2.1 Klasifikasi Tipe Kompresi Data

Secara umum, kompresi data dapat di klasifikasikan ke dalam 2 macam, yaitu :

1. Kompresi *Lossy*

   Teknik kompresi dimana data yang sudah dikompresi tidak dapat dikembalikan seperti data semula, dimana *lossy* atau *distortive* atau *noise-incurring*. Kompresi deperti ini digunakan untuk gambar dan suara dimana kehilangan *(loss)* data dapat diijinkan dalam kasus tertentu.

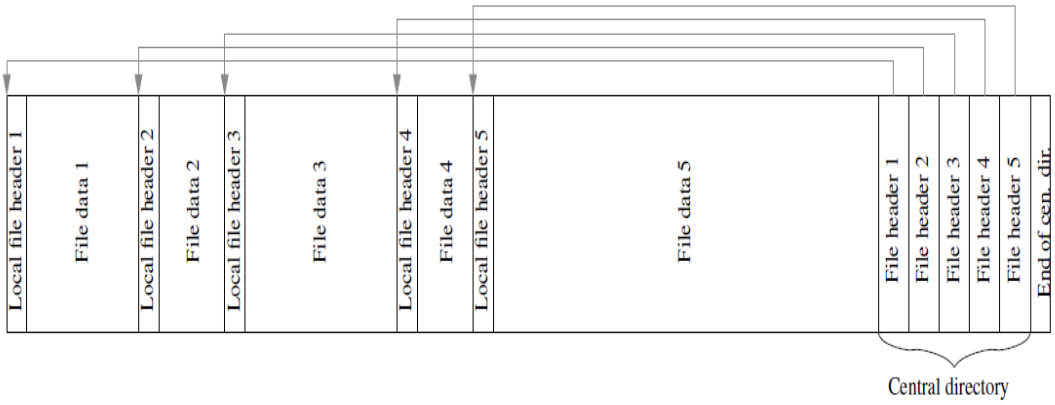
2. Kompresi *Lossless*

   Kompresi *lossless*/tanpa *distorsi*/tanpa *noise* adalah teknik kompresi untuk data seperti *file* progaram, *file* dokumen dan *record* basis data dimana sama sekali tidak diijinkan perbedaan antara awal (sebelum kompresi) dan data setelah dilakukan kompresi

2.3 Algoritma GZIP

GZip merupakan perkembangan dari algoritma Deflate yang merupakan kombinasi dari algoritma LZ77 dan algoritma Huffman [5].

Struktur GZip dapat dilihat pada gambar 2.6. Setiap *file* terdiri atas rangkaian dimana *header* memuat tentang informasi *file*, kompresi dan enkripsi. Lalu semua *file* akan dirangkai dalam sebuah *arbitrary* dan sebuah informasi mengenai keseluruhan GZip. Pada sebuah informasi, masing-masing memiliki *file-headers*, yang menggandung kedua informasi tentang *file* dan informasi tentang dimana GZip akan dimulai.



Gambar 2.6 Struktur file GZip

2.3.1 Algoritma LZ77

LZ77 merupakan algoritma kompresi data *lossless* yang dikembangkan oleh dua orang ilmuan komputer yang bernama Abraham Lempel dan Jacob Ziv pada tahun 1977. Inti dari algoritma LZ77 adalah *string matching* di mana sebuah *string* panjang dibagi menjadi 2 blok yaitu *Sliding Window* dan *Read Ahead* atau *Look Ahead* [6].

Pencarian kata (*string matching*) dilakukan dengan mencocokkan *string* pada *String Ahead* dengan *string* pada *Sliding Window*. Jika sama, maka dibentuk kode yang terdiri dari pasangan *offset* dan *length* dan akan disalin ke *file output.* Setelah itu indeks awal *Sliding Window* akan bergeser ke kanan sebanyak *lenght* adalah panjang karakter yang ditemukan sama dikurangi 1. Jika isi *Read Ahead* dukurangi 1. Jika masih tidak ditemukan kesamaan hingga panjang isi dari *Read Ahead* = 2 maka isi *Read Ahead* akan disalin ke *file output* dan indeks awal *Siling Window* akan bertambah 2. Berikut contoh penyelesaian suatu kasus dengan metode LZ77.

Contoh sebuah string : AAAAAAABCABCAAAAA

Sliding Window (merah) : AAAAAAABCA

Read Ahead (biru) : BCAAAAA

Karena isi *Read Ahead* tidak sama dengan *Sliding Window* maka indeks akhir *Read Ahead* akan dikurangi 1 menjadi BCAAAA. Karena masih tidak ditemukan kesamaan, maka indeks akhir *Read Ahead* akan dikurangi terus hingga menjadi :

Sliding Window : AAAAAAABCA

Read Ahead : BCA

Karena isi *Read Ahead* terhadap kesamaan dengan isi *Sliding Window* maka dicari nilai *offset* dan *length*-nya. Perhitungan sebagai berikut :

Offset = indeks akhir dari *Sliding Window* – indeks awal rangkaian yang sama

= 10 – 8 = 2

Length = panjang karakter yang sama -1

= 3 – 1 = 2

Selain itu diberikan juga *flag* untuk membedakan hasil dari kompresi yang mana yang merupakan karakter literal dan kode. Di mana *flag* 0 untuk karakter literal dan *flag* 1 untuk kode. Contohnya, hasil kompresi adalah AAAAAAAAAABCA2293. Di sini tampak jelas yang mana yang merupakan literal(huruf) dan kode(angka). Tetapi bagaimana jika hasil kompresi adalah 222222222223922293 maka akan sangat sulit membedakan yang mana literal dan yang mana kode. Oleh karena itu *flag* akan digunakan untuk menandai sebuah literal atau kode.

Contoh :

Hasil : 222222222223922293

Flag : 000000000000001111

Flag kemudian di potong tiap 8 *bit* hingga menjadi 1 *byte* yang akan digunakan sebagai *prefix code.* Contoh :

Flag : 00000000 00000011 11000000

Bagaimana yang diwarnai biru merupakan *padding* karena 1 blok harus terdiri dari 8 *bit* sedangkan blok tersebut hanya terdiri dari 2 *bit* saja. Dari *flag* di atas, maka akan dicari *prefix code* yang merupakan bentuk desimal dari deret biner yang dibentuk oleh *flag*. Maka *prefix code* yang duhasilkan adalah :

Flag : 00000000 00000011 11000000

Prefix code : 0 3 192

Prefix code akan diselipkan pada hasil kompresi sehingga menjadi :

0222222223222392221929 3

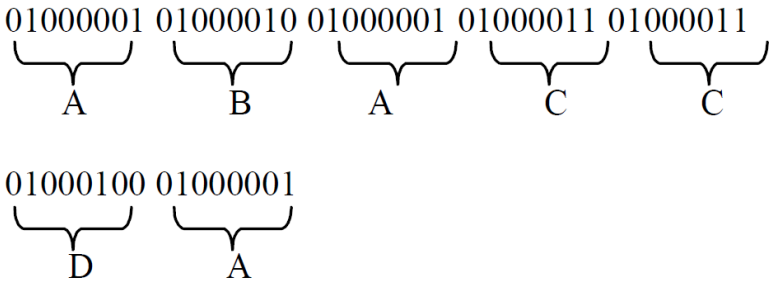
2.3.2 Algoritma Huffman

Algoritma Huffman ditemukan oleh David Huffman pada tahun 1952. Algoritma ini menggunakan pengkodean yang mirip dengan kode Morse [7]. Berdasarkan tipe kode yang digunakan algoritma Huffman termasuk metode statistic. Sedangkan berdasarkan teknik pengkodeannya menggunakan metode symbolwise. Algoritma Huffman merupakan salah satu algoritma yang digunakan untuk mengompres teks. Algoritma Huffman secara lengkap:

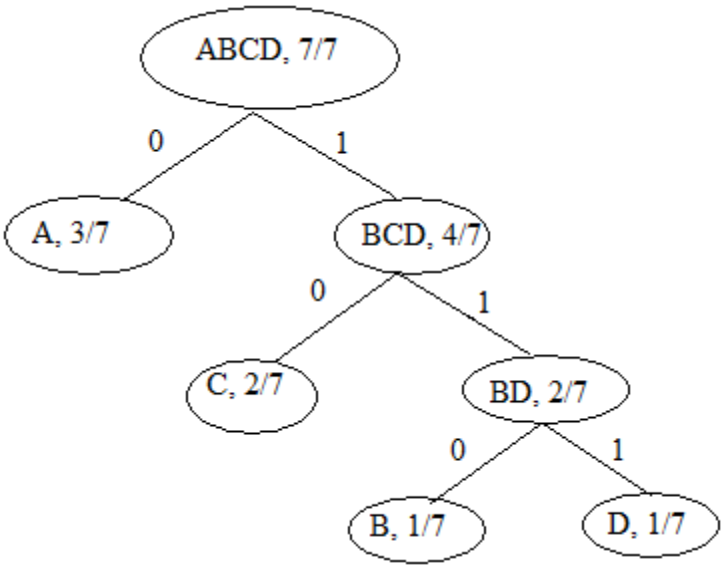
Terdapat *file* teks yg berisi ‘ABACCDA’

1. Pilih dua simbol dengan peluang (probability) paling kecil (pada contoh di atas simbol B dan D). Kedua simbol tadi dikombinasikan sebagai simpul orangtua dari simbol B dan D sehingga menjadi simbol BD dengan peluang 1/7 + 1/7 = 2/7, yaitu jumlah peluang kedua anaknya.
2. Selanjutnya, pilih dua simbol berikutnya, termasuk simbol baru, yang mempunyai peluang terkecil.
3. Ulangi langkah 1 dan 2 sampai seluruh simbol habis.

Sebagai contoh, dalam kode ASCII *string* 7 huruf "ABACCDA" membutuhkan representasi 7 × 8 bit = 56 bit (7 byte), dengan rincian sebagai berikut:



Untuk mengurangi jumlah bit yang dibutuhkan, panjang kode untuk tiap karakter dapat dipersingkat, terutama untuk karakter yang frekuensi kemunculannya besar. Pada *string* di atas, frekuensi kemunculan A = 3, B = 1, C = 2, dan D = 1, sehingga dengan menggunakan algoritma di atas diperoleh kode Huffman seperti pada Tabel 1.



Gambar 2.7 Pohon Huffman untuk "ABACCDA"

Tabel 2.1 Kode Huffman

Karakter	**Frekuensi**	**Peluang**	**Kode Huffman**
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111

Dengan menggunakan kode Huffman ini, *string* "ABACCDA" direpresentasikan menjadi rangkaian bit : 0 110 0 10 10 111 0. Jadi, jumlah bit yang dibutuhkan hanya 13 bit dari yang seharusnya dibutuhkan 56 bit.

Untuk menguraikan kembali data yang sudah dikodekan sebelumnya dengan algoritma *Huffman*, dapat digunakan cara sebagai berikut :

1. Baca *bit* pertama dari *string biner* masukan
2. Lakukan traversal pada pohon Huffman mulai dari akar sesuai dengan *bit* yang dibaca. Jika *bit* yang dibaca adalah 0 maka baca anak kiri, tetapi jika *bit* yang dibaca adalah 1 maka baca anak kanan.
3. Jika anak dari pohon bukan daun (simpul tanpa anak) maka baca *bit* berikutnya dari *string biner* masukan.
4. Hal ini diulang (traversal) hingga ditemukan daun.
5. Pada daun tersebut simbol ditemukan dan proses penguraian kode selesai.
6. Proses penguraian kode ini dilakukan hingga keseluruhan *string biner* masukan diproses.

2.3.3 Prosedur Algortima GZip

Algoritma LZ77 diubah adalah algoritma kompresi lossless. Metode kompresi didasarkan pada konsep bahwa *string* karakter dapat mencocokkan *string* identik karakter dalam *string* yang muncul sebelumnya. Karena itu, ini dikategorikan sebagai kompresi metode kamus. LZ77 diubah menggunakan ukuran jendela tetap dikenal sebagai jendela geser. Algoritma memungkinkan pencarian buffer dengan panjang sampai 32 KB. Jendela *buffer* pencarian memegang catatan karakter yang telah melewati *look-ahead buffer* dan tersedia untuk mencari untuk mencocokkan urutan baru karakter dalam *lookahead buffer*. Ketika urutan baru karakter, yang akan dikompresi, yang cocok dengan *string* karakter dalam *buffer* pencarian, *string* yang cocok digantikan oleh dua angka, juga dikenal sebagai *token*, dalam output terkompresi *stream*. Kedua *token* menunjukkan:

1. Yang pertama: Memberikan informasi tentang jarak, tentang seberapa jauh kembali ke pencarian *buffer* cocok *string* mulai urutan karakter.
2. Yang kedua: Memberikan informasi tentang panjang karakter string yang dicocokkan.

Karakter yang tidak cocok hanya ditulis ke kompresi *stream*, bukan sebuah token. Oleh karena itu, aliran kompresi terdiri dari tiga entitas:

1. Literal (karakter yang cocok/sama),
2. Distance (jarak offset yg cocok) dan
3. Length (panjang *string* yg cocok).

Dalam laporan ini, istilah untuk format ini adalah LDL – Literal Distance Length.

Output stream dikompresi dari algoritma LZ77 kemudian akan dikodekan dengan "statis" Huffman *coding* menggunakan total tiga tabel kode, dimana satu untuk *literal, distance dan length*.

Pengkodean yang digunakan oleh algoritma LZ77 beroperasi pada karakter ASCII, maka ukuran huruf adalah 256. Setelah itu dianalisa dan diperiksa berulang menggunakan Algoritma LZ77, kode baru dalam kode vektor v dibentuk setelah skema *literal-distance-length* (LDL). Jenis pengkodean Huffman yang dalam laporan ini adalah statis Huffman *coding*. Istilah "statis" berasal dari kenyataan bahwa *codewords* Huffman tidak mengubah seluruh file, yang dikodekan menggunakan algoritma LZ77. Di sisi lain "dinamis" Huffman coding adalah kode Huffman yang dibuat untuk masing-masing panjang blok yang digunakan oleh algoritma LZ77. Setelah setiap blok telah diperintahkan oleh

format LDL. Dalam *file* GZip "statis" Huffman-*encode*, standar yang sudah mendefinisikan tabel Huffman, dan sudah diimplementasikan dalam berbagai alat GZip. Untuk *file* GZip "dinamis" Huffman-*encoded*, struktur pohon harus dikirim sebagai blok yang terpisah, yang menunjukkan pemetaan antara *codewords* Huffman dan *literal* (alfabet). Pemetaan antara LDL dan kode Huffman adalah sebagai ditunjukkan pada Tabel 2.1. Seperti disebutkan, 256 pertama *literal* menunjukkan karakter ASCII.

Tabel 2.2 Pencatatan codewords prefiks Huffman statis untuk semua LDL

Prefix Huffman codes for all LDLs		
Literal	#Bits	Prefix Huffman code range
0-143	8	00110000-10111111
144-255	9	110010000-111111111
256-279	7	0000000-0010111
280-287	8	11000000-11000111

Ini diwakili tanpa membubuhkan bit tambahan, yaitu LDL dalam kisaran ini benar-benar didefinisikan oleh *prefiks* Huffman *codewords*. *Literal* 257-285 berdedikasi untuk menunjukkan panjang dari string yang cocok, sementara 286-287 tidak pernah digunakan. Tergantung pada panjang simbol yang sesuai, kode *affix* mungkin diperlukan, sesuai dengan Tabel 2.2. Misalnya, indikasi panjang string simbol dari buffer pencarian yang muncul di blok saat ini dapat memiliki panjang 120. Hal ini akan menghasilkan *codeword* Huffman 11000000|0110, di mana "|" menunjukkan pemisahan antara awalan dan *affix* bit.

Tabel 2.3 Pencatatan affix codewords Huffman statis untuk semua indikasi panjang

Huffman codeword affixes for lengths								
LDL	Extra Bits	Length	LDL	Extra Bits	Length	LDL	Extra Bits	Length
257	0	3	267	1	15, 16	277	4	67-82
258	0	4	268	1	17, 18	278	4	83-98
259	0	5	269	2	19-22	279	4	99-114
260	0	6	270	2	23-26	280	4	115-130
261	0	7	271	2	27-30	281	5	131-162
262	0	8	272	2	31-34	282	5	163-194
263	0	9	273	3	35-42	283	5	192-226
264	0	10	274	3	43-50	284	5	227-257
265	1	11, 12	275	3	51-58	285	0	258
266	1	13, 14	276	3	59-66			

Kode bit terakhir ditambahkan dalam format GZip adalah kode jarak, yang terdaftar dalam Tabel 2.3. Sebagaimana dicatat, membubuhkan bit juga diperlukan tergantung pada jarak. Sebuah contoh penggunaan Tabel 1.3 sekarang dibuat A jarak 65 dalam string teks non-encoded, dikodekan sebagai 01100|00000, sedangkan jarak dari 67 hasilnya 01100|00010. Di sini, "|" menunjukkan pemisahan antara kode Huffman dan bit ekstra.

Tabel 2.4 Pencatatan codewords Huffman statis untuk semua indikasi jarak.

Huffman codewords for distances								
Codeword	Extra Bits	Distance	Codeword	Extra Bits	Distance	Codeword	Extra Bits	Distance
0	0	1	10	4	33-48	20	9	1025-1536
1	0	2	11	4	49-64	21	9	1537-2048
2	0	3	12	5	65-96	22	10	2049-3072
3	0	4	13	5	97-128	23	10	2073-4096
4	1	5, 6	14	6	129-192	24	11	2097-6144
5	1	7, 8	15	6	193-256	25	11	6145-8192
6	2	9-12	16	7	257-384	26	12	8193-12288
7	2	13-16	17	7	385-512	27	12	12289-16384
8	3	17-24	18	8	513-768	28	13	16385-24576
9	3	15-32	19	8	769-1024	29	13	24577-32768

2.4 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Dikembangkan oleh Sun Microsystems dan diterbitkan tahun 1995 [10]. Java tidak boleh disalahpahami sebagai JavaScript. JavaScript adalah bahasa scripting yang digunakan oleh web browser.

2.4.1 Sejarah Perkembangan Java

Bahasa pemrograman Java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot *Duke* yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran *Sand Hill Road* di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program *Java Oak* pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan TV kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan

pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.

2.4.2 Keunggulan Java

Multiplatform. Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip *tulis sekali, jalankan di*

mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows. Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk meninterpretasikan *bytecode* tersebut.

OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut *Object*. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan Java sebagai salah satu bahasa pemograman termudah, bahkan untuk fungsi fungsi yang advance seperti komunikasi antara komputer sekalipun.

Perpustakaan Kelas Yang Lengkap, Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus

membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.

Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.4.3 Kelemahan Java

a. Tulis sekali, perbaiki di mana saja - Masih ada beberapa hal yang tidak kompatibel antara *platform* satu dengan platform lain. Untuk J2SE, misalnya SWT-AWT bridge yang sampai sekarang tidak berfungsi pada Mac OS X.
b. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma yang digunakan

program akan lebih sulit disembunyikan dan mudah dibajak/di*reverse-engineer*.

c. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.5 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*) [11]. Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM Visual Age for Java 4.0. Produk ini diluncurkan oleh IBM pada tanggal 5 November 2001.

Keunggulan:

1. **Multi-platform**: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. **Mulit-language**: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.

3. **Multi-role**: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.