BAB II

TINJAUAN PUSTAKA

II.1. Aplikasi

Aplikasi adalah suatu *sub* kelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yagn diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi sutau pake disebut sebagai suatu paket atau *suite* aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice.org*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki atarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah. (*Dahlan Abdullah ; 2013 : 152*)

Jenis-jenis *Software* Aplikasi :

1. *Software* aplikasi hiburan, contohnya yaitu winamp untuk mendengarkan musik, *games* dan sebagainya untuk hiburan.
2. *Software* aplikasi pendidikan yaitu *software* digunakan untuk mempelajari atau mereferensikan tentang pendidikan atau pengetahuan.

3. *Software* aplikasi bisnis yaitu *software* yang digunakan untuk aplikasi bisnis
4. *Software* aplikasi khusus
5. *Software* aplikasi untuk produktivitas kerja.

II.2. Jaringan

Pengertian jaringan komputer merupakan sekumpulan komputer serta perangkat-perangkat lain pendukung komputer yang saling terhubung dalam suatu kesatuan. Media jaringan komputer dapat melalui kabel-kabel atau tanpa kabel sehingga memungkinkan pengguna jaringan komputer dapat saling melakukan pertukaran informasi seperti dokumen dan data. Dapat juga melakukan pencetakan pada *printer* yang sama dan bersama-sama memakai perangkat keras dan perangkat lunak yang terhubung dengan jaringan. (*Choirul Muallifah ; 2013 : 2*)

II.2.1. Jenis-Jenis Jaringan

Berdasarkan jangkauan area atau lokasi jaringan menurut *Choirul Muallifah* tahun 2013 dibedakan menjadi 3 jenis, yaitu :

1. LAN (*Local Area Network*)

Merupakan jaringan lokal yang dibuat pada area tertutup. Misalkan dalam satu gedung atau dalam satu ruangan. LAN biasa digunakan untuk jaringan kecil yang menggunakan *resource* bersama. Seperti penggunaan *printer* secara bersama. LAN dapat menggunakan media komunikasi seperti kabel dan *wireless*.

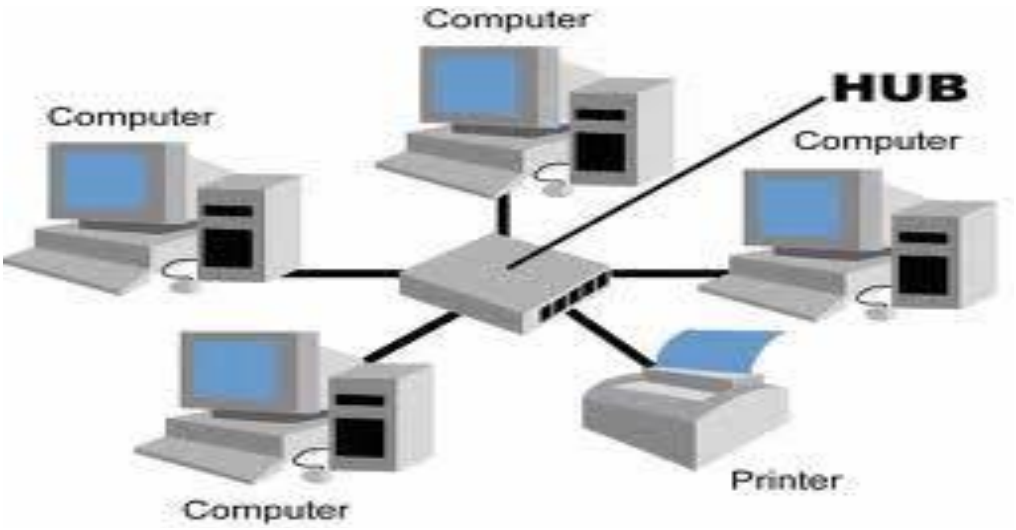
a. Komponen Jaringan LAN

Ada dua hal utama yang harus dipertimbangkan ketika merencanakan atau memasang LAN, yaitu komponen *hardware* jaringan dan *software* jaringan . Ada 3 kategori utama peralatan yang membentuk komponen *hardware* dari jaringan local. Ketiga kategori tersebut adalah:

a) *Server*
b) Sistem Komunikasi LAN
c) *Workstation*

b. Cara Kerja Jaringan LAN

Jaringan LAN umumnya menggunakan *Switch Hub*. Kemudian *Hub* akan mengikuti prinsip kerja hub itu sendiri. Dalam hal ini biasanya salah satu komputer diantara jaringan komputer itu akan digunakan menjadi *server* yang mengatur semua sistem didalam jaringan tersebut. Sehingga jaringan bisa terhubung dengan data lain bisa kerjasama. Berikut sistem kerja jaringan:



Gambar II.1. Sistem Kerja Jaringan
Sumber : Choirul Muallifah ; 2013 : 2

c. Topologi Jaringan LAN

Topologi jaringan merupakan gambaran pola hubungan antara komponen-komponen jaringan, yang meliputi komputer *server*, komputer *client* atau *workstation*, *hub* atau *switch*, pengkabelan, dan komponen jaringan yang lain.

a) Topologi Bus

Merupakan topologi yang menghubungkan beberapa *computer* ke sebuah kabel dengan beberapa terminal. Menggunakan jenis kabel *Coaxial* dengan konektor BNC.

b) Topologi Token *Ring*

Merupakan penyempurnaan dari topologi *Ring*. Pada topologi ini *collions* (tumbukan antar pengiriman data) dapat dicegah.

c) Topologi *Star*

Merupakan topologi yang menghubungkan beberapa *computer* dengan menggunakan perangkat yaitu *Hub* atau *Switch*.

2. MAN (*Metropolitan Area Network*)

Merupakan jaringan antara LAN satu dengan LAN lain yang dipisahkan daerah lokasi yang cukup jauh. Contoh penggunaan MAN adalah hubungan antara kantor pusat dengan kantor cabang yang ada di daerah-daerah. Dapat dikatakan MAN merupakan pengembangan dari LAN.

3. WAN (*Wide Area Network*)

Merupakan jaringan yang cakupannya lebih luas dari pada MAN. Cakupan WAN meliputi satu kawasan, satu negara, satu pulau, bahkan satu benua. Metode yang digunakan WAN hampir sama dengan LAN dan MAN.

II.2.2. Manfaat dan Keuntungan Jaringan

Adapun manfaat dan keuntungan membangun jaringan menurut *Choirul Muallifah* tahun 2013 adalah sebagai berikut :

1. Manfaat Jaringan Komputer:
    a. *Sharing resources*
    b. Media komunikasi
    c. Integrasi data
    d. Pengembangan dan pemeliharaan
    e. Keamanan data
    f. Sumber daya lebih efisien dan informasi terkini
2. Keuntungan Membangun Jaringan Komputer Yaitu:
    a. Dapat berbagi peralatan (*peripheral*) dan penggunaanya, seperti *printer*, *harddisk, modem*.
    b. Memudahkan bertukar data diantara pengguna komputer tanpa harus menggunakan disket.
    c. Kita dapat menggunakan programprogram yang ada di komputer pusat.
    d. Bisa mengirim dan menerima *email* dari *internet* dan mencari informasi lain melalui fasilitas *internet*.

II.3. *Monitoring* Jaringan

Manajemen jaringan adalah kemampuan untuk memonitor, mengontrol, dan merencanakan suatu jaringan komputer dan komponen sistem. *Monitoring*

jaringan merupakan bagian dari manajemen jaringan. Hal yang paling mendasar dalam konsep manajemen jaringan adalah tentang adanya manajer atau perangkat yang memanajemen dan agen atau perangkat yang dimanajemen. Dalam implementasinya, ada berbagai macam arsitektur manajemen jaringan yang didasarkan pada tipe dan ukuran masing-masing. Ada dua arsitektur yang dapat digunakan yaitu manajemen terpusat (*centralized management*) dan manajemen menyebar (*distributed management*). (*Reza Pradikta ; 2013 : A-154*)

Konsep *Network Monitoring System* (NMS) sebenarnya sederhana yaitu sistem ekstra atau kumpulan sistem yang memiliki tugas mengamati atau memonitor sistem-sistem terhadap kemungkinan terjadinya masalah-masalah pada sistem tersebut untuk dapat dideteksi secara dini. Sebagai contoh, suatu *monitoring* sistem dapat secara periodik menghubungi suatu *web server* untuk menjamin adanya respon dari *web server*, jika tidak ada respon maka *monitoring system* kemudian mengirimkan pesan atau notifikasi ke administrator. Hal – hal yang bakal dimonitoring dalam *network* tentunya akan sangat kompleks, dan sistem *monitoring* yang baik seharusnya menyediakan *history* dan *log* yang memungkinkan kita membuat laporan, statistik dan *graph* dari masing - masing *object* yang dimonitoring sehingga sistem NMS yang digunakan memberikan kontribusi penuh dalam pendeteksian secara dini terhadap kemungkinan masalah - masalah yang timbul.(*Nurul Fatmawati Asri ; 2014 : 152*)

Nagios merupakan sebuah sistem dan aplikasi monitoring jaringan yang diciptakan oleh Ethan. *Nagios* mengawasi *host-host* dan *service* yang telah ditetapkan, memberi peringatan jika keadaan memburuk, dan memberi tahu kapan

keadaan tersebut membaik. Awalnya *Nagios* didesain hanya dapat dijalankan pada sistem operasi *Linux*, namun sekarang *Nagios* dapat berjalan dihampir semua sistem operasi berbasis *Unix Nagios* memiliki beberapa fitur. (*Nurul Fatmawati Asri ; 2014 : 152*) di antaranya adalah sebagai berikut:

1. Memonitoring servis jaringan (SMTP, POP3, HTTP, NNTP, PING, dsb)
2. Memonitoring sumber-sumber *host* (*load* prosesor, penggunaan disk, dsb)
3. *Desain plugin* yang serderhana, yang mengijinkan pengguna untuk lebih mudah menggunakan pemeriksaan terhadap servisnya.

II.4. *Sniffer*

Sniffer adalah program yang membaca dan menganalisa setiap protokol yang melewati mesin di mana program tersebut diinstal. Secara *default*, sebuah komputer dalam jaringan (*workstation*) hanya mendengarkan dan merespon paket-paket yang dikirimkan kepada mereka. Namun demikian, kartu jaringan *(network card)* dapat diset oleh beberapa program tertentu, sehingga dapat memonitor dan menangkap semua lalu lintas jaringan yang lewat tanpa peduli kepada siapa paket tersebut dikirimkan. Aktifitasnya biasa disebut dengan sniffing.

Untuk dapat membaca dan menganalisa setiap protokol yang melewati mesin, diperlukan program yang bisa membelokkan paket ke komputer *attacker*. Biasa disebut serangan *spoofing*. *Attacker* akan bertindak sebagai *Man-In-the-Middle* (MIM).

II.4.1. Kegunaan dan Kerugian *Sniffer*

1. Kegunaan dari sniffer adalah sebagai berikut :
    a. Menangkap *password clear text* dan nama *login* dari jaringan.
    b. Konversi data jaringan ke bentuk yang mudah dipahami manusia.
    c. *Fault analysis* untuk menemukan permasalahan-permasalahan dalam jaringan.
    d. *Performance analysis* untuk menemukan *bottleneck* dalam jaringan.
    e. *Network intrusion detection* untuk menemukan *hacker/cracker*.
    f. *Network traffic logging*, untuk membuat *log* yang tidak dimodifikasi dan dihapus oleh *hacker*.
2. Kerugian dari *Sniffer* sebagai berikut :
    a. *Sniffer* dapat mencuri *password*.
    b. *Sniffer* dapat mendapatkan rahasia atau informasi eksklusif.
    c. Mereka dapat digunakan untuk pelanggaran keamanan jaringan tetangga, atau mendapatkan akses *leveraged*.

II.4.2. Macam – Macam *Sniffer*

a. *tcpdump*. Program *wiretap* terumum dan tertua. Dalam mode yang paling sederhana,ia akan menghasilkan satu baris dekode paket-paket ke *commandline*,satu baris per paket. *tcpdump* merupakan program penangkap paket standar *UNIX*. Dapat diperoleh di *http://www.tcpdump.org/*.
b. *Ethereal.* Merupakan program *sniffing* berbasis GUI terbaik untuk UNIX.

Tersedia di http://ethereal.zing.org

c. *sniffit* (http://reptile.rug.ac.be/~coder/sniffit/sniffit.html) Berguna ketika berusaha menganalisis data *layer* aplikasi.

d. *Snort.Packet-sniffer/logger* berbasis *libcap* dengan *filtering* yang ekstensif.Dapat diperoleh dihttp://www.clark.net/~roesch/security.html

e. *trinux*.Berisikan *tcpdump* dan *sniffit* dalam satu buah *floppy bootable disk*.Tersedia di http://www.trinux.org/

f. *karpski*(http://niteowl.userfriendly.net/linux/RPM/karpski.html).

g. Program paket *sniffer* GUI.

h. *SuperSnifferv1.3*(http://www.mobis.com/~ajax/projects/).Merupakan *sniffer* paket berbasis *libpcap* yang diperbaiki dengan banyak modifikasi seperti *file log* yang terenkripsi DES, lalu lintas dapat di*log* dengan pencocokan pola oleh ekspresi reguler, koneksi POP dan FTP d*ilog* pada satu baris, *telnet negotiation garbage* diabaikan, pengabaian koneksi ganda, *tcp packet reassembly, parellel tcp connection logging.*

i. *exdump* (http://exscan.netpedia.net/exdump.html). *Lightweight packet sniffer* untuk *Linux?*

j. *linux_sniffer.c*. Program ini terdiri dari 175 baris kode bahasa C, didistribusikan utamanya pada *site-site cracker* di *Internet*. Program ini spesifik *Linux*. Bersifat *free* dan merupakan cara yang mudah mempelajari lalu lintas paket.

II.4.3. Alat – Alat Untuk Mendeteksi *Sniffer*

1. *IFCONFIG*

   Secara *default,workstation* mendengarkan dan menanggapi hanya paket-paket yang dialamatkan padanya. Namun, bila *network interface workstation* dialihkan ke mode *promiscuous, workstation* dapat memonitor dan menangkap seluruh lalu lintas jaringan dan paket-paket yang lewat, tanpa memperdulikan tujuannya.

2. IFSTATUS

   Ifstatus memeriksa seluruh *interface* jaringan pada sistem dan melaporkan *interface* yang berada pada mode *debug* atau *promiscuous.*

3. ANTISNIFF

   Merupakan alat pendeteksi *sniffer* yang paling *komprehensif*.

4. NEPED(Network Promiscuous Ethernet Detector)

   Sebuah *tool* dari *The Apostols* yang dapat mendeteksi *sniffer* yang berjalan pada segmen lokal.

II.5. Data dan Packet Data

Pengertian Data dan Packet Data dalam jaringan pengertian data adalah : Kumpulan file atau informasi dengan tipe tertentu, baik berupa suara, gambar, atau yang lainnya (berupa *text*). Pengertian packet data (dalam jaringan) : Paket jaringan atau network packet adalah satuan informasi dasar yang dapat ditransmisikan di atas jaringan atau melalui saluran komunikasi digital. Sebuah

paket berisi *packet header* yang berisi informasi mengenai protokol tersebut (informasi mengenai jenis, sumber, tujuan, atau informasi lainnya), data yang hendak ditransmisikan yang disebut dengan data *payload*, dan *packet trailer* yang bersifat opsional. Sebuah paket memiliki struktur logis yang dibentuk oleh protokol yang digunakannya. Ukuran setiap paket juga dapat bervariasi, tergantung struktur yang dibentuk oleh arsitektur jaringan yang digunakan. Paket jaringan juga dapat disebut datagram, *frame*, atau *cell*.

Pengertian *bandwidth* adalah suatu ukuran dari banyaknya informasi yang dapat mengalir dari suatu tempat ke tempat lain dalam suatu waktu tertentu. *Bandwidth* dapat dipakai untuk mengukur baik aliran data analog maupun aliran data digital. Satuan yang digunakan untuk *Bandwdth* adalah bps *(bit per second)*. Satuan ini berarti jumlah *bit* yang dapat mengalir tiap detik melalui suatu media. Seperti yang kita ketahui *bit (binary digit)* hanya terdiri dari dua angka yaitu 0 dan 1. Konsep *bandwidth* juga bergantung pada media dan jarak yang digunakan untuk mengalirkan data dalam jaringan. Konsep *bandwidth* ini tentu saja juga mempunyai kelemahan, *bandwidth* tidak dapat menghitung berdasarkan kondisi jaringan yang sebenarnya. *Bandwidth* atau sering disebut juga *throughput* adalah lebar *"width"* dari sebuah *frequency band. Bandwidth* dari sebuah jaringan ditentukan dari jumlah *bit* yang dapat di transmisikan melalui jaringan*/network* dalam periode waktu tertentu. Misalnya sebuah jaringan memiliki *bandwidth* sebesar 10 Mbps, hal ini berarti bahwa jaringan tersebut mampu untuk mengirimkan 10 juta bit tiap detiknya.

Troughput adalah *bandwidth* yang sebenarnya (aktual) yang diukur dengan satuan waktu tertentu dan pada kondisi jaringan tertentu yang digunakan untuk melakukan transfer file dengan ukuran tertentu, perbandingan *bandwidth* dan *throughput* yaitu waktu download terbaik adalah ukuran file dibagi dengan *bandwidth*. Sedangkan waktu aktual atau sebenarnya adalah ukuran file dibagi dengan *throughput*. *Bandwidth* berbeda dengan kecepatan.

Kecepatan adalah jarak yang ditempuh dalam satu satuan waktu, sedangkan *bandwidth* adalah banyaknya *bit* yang dapat dikirimkan per detik. Semakin banyak *bit* yang dapat dikirim dalam satuan waktu misalnya detik, berarti lebih besar *bandwidth* nya sehingga menjadi lebih cepat kecepatan transfer datanya. Berarti secara tidak langsung dapat diambil kesimpulan bahwa *bandwidth* lah yang mempengaruhi besarnya kecepatan transfer data di dalam jaringan.

Protokol adalah sebuah aturan atau standar yang mengatur atau mengijinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer. Protokol dapat diterapkan pada perangkat keras, perangkat lunak atau kombinasi dari keduanya. Pada tingkatan yang terendah, protokol mendefinisikan koneksi perangkat keras. Secara umum fungsi protokol adalah sebagai penghubung dalam komunikasi data sehingga proses penukaran data bisa berjalan dengan baik dan benar.

II.5.1. Fungsi Protokol

Secara khusus, fungsi protokol adalah sebagai berikut :

a) *Fragmentasi* dan *Re-assembly* Pembagian informasi yang dikirim menjadi beberapa paket data dari sisi pengirim. Jika telah sampai di penerima, paket data tersebut akan digabungkan menjadi paket berita yang lengkap.
b) Enkapsulasi (*Encaptulation*) adalah proses pengiriman data yang dilengkapi dengan alamat, kode-kode koreksi, dan lain-lain.
c) *Flow Control* Fungsi dari *Flow Control* adalah sebagai pengatur jalannya data dari pengirim ke penerima. *e.Error Control* Tugasnya adalah mengontrol terjadinya kesalahan sewaktu data dikirimkan.
d) Pelayanan Transmisi Fungsinya adalah memberikan pelayanan komunikasi data yang berhubungan dengan prioritas dan keamanan data.

II.5.2. Jenis – Jenis Protokol

Ada beberapa macam protokol pada suatu jaringan yang meliputi, ARP, ICMP, DHCP, DNS, IP, TCP, UDP, HTTP, FTP, SMTP, POP, IMAP, WLAN (IEEE 802.11), dan TLS/SSL.

a) *Address Resolution Protocol* (ARP) adalah protokol untuk *mapping* dari alamat IP *(Internet Protocol)* ke alamat fisik MAC (*Media Access Control*). Misal di suatu jaringan kita ingin mengirim paket ke host A 192.168.1.2, maka pertama kita harus tahu siapa yg mempunyai alamat IP tersebut. Maka ARP akan menjawab pertanyaan tersebut ke semua *host* yang ada di jaringan. Maka alamat IP tersebut akan menjawab kembali jawaban tersebut dengan mengirimkan alamat MACnya. Alamat MAC ini akan disimpan di tabel ARP untuk memudahkan pencarian jika diperlukan

pengiriman paket ke tujuan yang sama. Dalam kasus kasus penjelajahan situs internet (misal, www.google.com) maka proses transmisi ARP harus dilakukan terlebih dahulu sebelum proses transmisi HTTP dimulai.

b) *Internet Message Controll Protocol* (IMCP). ICMP merupakan protokol pelengkap dalam IP *(Internet Protocol)*. Seperti halnya IP, ICMP bekerja pada *network layer* pada susunan OSI *Layer*. ICMP di desain untuk mengontrol pengiriman dan pesan percobaan melewati jaringan IP. Kemampuan untuk memahami ICMP adalah sangat di dibutuhkan untuk setiap perangkat *network* yang kompatibel dengan IP. Bagaimanapun juga banyak perangkat keamanan seperti *firewall* memblok atau me-non aktifkan semua bagian dari fungsi ICMP untuk kepentingan keamanan.

c) *Dynamic Configuration Protocol* (DHCP) adalah layanan yang secara otomatis memberikan nomo`r IP kepada komputer yang memintanya. Komputer yang memberikan nomor IP disebut sebagai DHCP *server*, sedangkan komputer yang meminta nomor IP disebut sebagai DHCP *client*. Dengan demikian *administrator* tidak perlu lagi harus memberikan nomor IP secara manual pada saat konfigurasi TCP/IP, tapi cukup dengan memberikan referensi kepada DHCP *server*.

d) *Domain Name Server*(DNS), yaitu *server* yang digunakan untuk mengetahui IP Address suatu *host* lewat *host name*. Dalam dunia *internet*, komputer berkomunikasi satu sama lain dengan mengenali IP *address*. Namun bagi kita tidak mungkin menghafalkan IP *address* tersebut, manusia lebih mudah menghapalkan kata-kata seperti www.yahoo.com,

www.google.com, atau www.facebook.com. DNS berfungsi untuk mengkonversi nama yang bisa terbaca oleh manusia ke dalam IP *address host* yang bersangkutan untuk dihubungi.

e) Internet protocol(IP) yang berperan dalam pentransmisian paket data dari *node* ke *node*. IP mendahului setiap paket data berdasarkan 4 *byte* (untuk versi IPv4) alamat tujuan (nomor IP). *Internet authorities* menciptakan *range* angka untuk organisasi yang berbeda. Organisasi menciptakan grup dengan nomornya untuk departemen. IP bekerja pada mesin *gateaway* yang memindahkan data dari departemen ke organisasi kemudian ke *region* dan kemudian ke seluruh dunia.

f) *Transmission transfer protocol*(TCP) berperan didalam memperbaiki pengiriman data yang benar dari suatu klien ke *server*. Data dapat hilang di tengah-tengah jaringan. TCP dapat mendeteksi *error* atau data yang hilang dan kemudian melakukan transmisi ulang sampai data diterima dengan benar dan lengkap.

g) *User Datagram Protocol (UDP)*, adalah TCP yang *connectionless*. Hal ini berarti bahwa suatu paket yang dikirim melalui jaringan dan mencapai komputer lain tanpa membuat suatu koneksi. Sehingga dalam perjalanan ke tujuan paket dapat hilang karena tidak ada koneksi langsung antara kedua host, jadi UDP sifatnya tidak *realibel*, tetapi UDP adalah lebih cepat dari pada TCP karena tidak membutuhkan koneksi langsung.

h) *Hypertext Transfer Protocol* (HTTP) adalah sistem untuk transmisi dan menerima informasi di *internet*. Http berfungsi sebagai permintaan dan

prosedur respon yang semua agen di Internet mengikuti sehingga informasi dapat cepat, mudah, dan akurat disebarluaskan antara *server*, yang memegang informasi, dan klien, yang mencoba untuk mengaksesnya. Http umumnya digunakan untuk mengakses halaman html, tetapi sumber daya lain bisa dimanfaatkan juga melalui http. Dalam banyak kasus, klien dapat bertukar informasi rahasia dengan server, yang perlu diamankan untuk mencegah akses yang tidak sah. Untuk alasan ini, https, atau http yang aman, dikembangkan oleh *netscape* untuk memungkinkan transaksi perusahaan otorisasi dan aman.

i) *File Transfer Protocol(*FTP*)* adalah suatu protokol yang berfungsi untuk tukar-menukar *file* dalam suatu network yang memberi *support* TCP/IP protokol. Dua hal penting yang ada dalam FTP adalah FTP *server* dan FTP *client*. FTP *server* menjalankan *software* yang digunakan untuk tukar menukar *file,* yang selalu siap memberian layanan FTP apabila mendapat *request* dari FTP *client*. FTP client adalah komputer yang menerimat koneksi ke FTP *server* untuk tujuan tukar menukar *file.*

j) Simple Mail Transfer Protocol(SMTP) adalah suatu protokol yang digunakan untuk mengirimkan pesan *e-mail* antar *server*, yang bisa dianalogikan sebagai kantor pos. Ketika kita mengirim sebuah *e-mail*, komputer kita akan mengarahkan *e-mail* tersebut ke sebuah SMTP *server*, untuk diteruskan ke *mail-server* tujuan. *Mail-server* tujuan ini bisa dianalogikan sebagai kotak pos di pagar depan rumah kita, atau kotak PO BOX di kantor pos. *Email-email* yang terkirim akan "nongkrong" di

tempat tersebut hingga si pemiliknya mengambilnya. Urusan pengambilan *e-mail* tersebut tergantung kapan di penerima memeriksa *account e-mailnya.*

k) *Post Office Protocol version* 3(POP3) adalah suatu protokol yang berfungsi untuk menarik atau mengambil *email* dari *server email* yang digunakan.

   Untuk menggunakan POP3 bisa dari *Microsoft Outlook*. biasanya untuk menggunakan POP3 di perlukan *settingan*:

l) *Internet Message Access Protocol*(IMAP) adalah protokol mail yang digunakan untuk mengakses *email* pada *web server remote* dari klien lokal. IMAP dan POP3 adalah dua yang paling umum protokol *internet mail* yang digunakan untuk mengambil *email*. Kedua protokol yang didukung oleh semua klien *email modern* dan *web server*.

m) WLAN (IEEE 802.11),

n) TLS/SSL, *Protocol* SSL dan TLS berjalan pada *layer* dibawah *application protocol* seperti HTTP, SMTP and NNTP dan di atas *layer* TCP *transport protocol*, yang juga merupakan bagian dari TCP/IP *protocol*. Selama SSL dan TLS dapat menambahkan keamanan ke protocol apa saja yang menggunakan TCP, keduanya terdapat paling sering pada metode akses HTTPS. HTTPS menyediakan keamanan *web-pages* untuk aplikasi seperti pada *Electronic commerce*. Protocol SSL dan TLS menggunakan *cryptography public-key* dan sertifikat *publik key* untuk memastikan identitas dari pihak yang dimaksud. Sejalan dengan peningkatan jumlah

client dan *server* yang dapat mendukung TLS atau SSL alami, dan beberapa masih belum mendukung. Dalam hal ini, pengguna dari *server* atau *client* dapat menggunakan produk *standalone*-SSL seperti halnya *stunnel* untuk menyediakan enkripsi SSL.

II.6. *Java*

Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan *Java* 2 adalah generasi kedua dari *Java platform* (generasi awalnya adalah JDK atau *Java Development Kit*). *Java* inilah yang berdiri diatas mesin *interpreter* yang diberi nama *Java Virtual Machine* (JVM). JVM inilah yang akan membaca *bytecode* dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin". Oleh karena itu bahasa java disebut juga sebagai bahasa pemrograman yang *portable* karena dapat dijalankan sebagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM. (*Utomo Budiyanto ; 2011 : 27*)

Sun Microsystems telah mendefinisikan tiga *platform java* menurut *Utomo Budiyanto 2011* yang masing – masing diarahkan untuk tujuan tertentu dan untuk lingkungan komputasi yang berbeda-beda:

1. *Java Standard Edition* (J2SE), adalah inti dari bahasa pemrograman *java*. JDK adalah salah satu *tool* dari J2SE untuk mengkompilasi program *java* pada JRE.

2. *Java Enterprise Edition* (J2EE), dengan *built-in* mendukung untuk *servlets*, JSP, dan XML, edisi ini ditujukan untuk aplikasi berbasis *server*.
3. *Java Micro Edition* (J2ME), didesain untuk meletakkan perangkat lunak *java* padabarang elektronik beserta perangkat pendukungnya.

II.6.1. J2ME (*Java 2 Micro Edition*)

Java Micro Editon atau yang biasa disebut J2ME adalah bagian dari J2SE, karena itu banyak pustaka yang ada pada J2SE dapat digunakan pada J2ME. Tetapi J2ME mempunyai beberapa pustaka khusus yang tidak dimiliki J2SE. Kelahiran *platform* J2ME timbul karena dibutuhkan adanya sebuah *platform* komputasi yang mengakomodasi piranti komputer elektronik dan *embedded*. Piranti ini dikelompokkan menjadi dua kategori, yaitu :

1. Personal, *piranti mobile* yang dapat digunakan untuk komunikasi melalui jaringan tertentu misalkan ponsel, Personal *Digital Assistant* (PDA), *Palm*, *Pocket* PC dan *organizer*.
2. Piranti informasi yang digunakan bersama dengan jaringan tetap, koneksi jaringan yang tidak putus-putus misalnya TV, *internet* dan sistem navigasi.

Kategori pertama mengarahkan piranti untuk tujuan khusus atau fungsi-fungsi tertentu yang terbatas dan tidak digunakan untuk mesin komputasi yang serba guna. Kategori kedua diarahkan untuk piranti yang mempunyai kapabilitas yang lebih besar dengan fasilitas *user interface* yang lebih baik, kemampuan komputasi yang lebih besar.

1.Keunggulan J2ME

Salah satu kelebihan *Java* yang paling signifikan adalah *run everywhere*. Dengan kelebihan ini, para pengembang yang sudah terbiasa mengembangkan aplikasi dalam bingkai kerja J2ME dan J2EE akan mampu bermigrasi dengan mudah untuk mengembangkan aplikasi J2ME. Selain itu, *Java* juga merupakan *platform* yang memiliki banyak keunggulan lain, keunggulan *Java* secara umum adalah :

a. *Multiplatform*, aplikasi J2ME bisa berjalan diatas banyak *platform* yang didalamnya terdapat JVM. Beberapa *platform* yang tersedia didalamnya terdapat JVM antara lain *Windows* CR, *Symbian, Embedded Linux* dan sebagainya.
b. *Robust*, kode-kode *Java* adalah kode-kode *robust*, karena *virtual machine* mengatur keamanan proses eksekusi aplikasi. *Java virtual machine* menyediakan *garbage collector* yang berfungsi mencegah kebocoran memory.
c. Terintegrasi dengan baik, J2ME bisa terhubung dengan *back-end* J2EE *server* dan *web services* dengan mudah karena menyediakan pustaka-pustaka API RMI dan *web services*.
d. Berorientasi obyek, *Java* merupakan salah satu bahasa pemrograman yang murni berorientasi obyek. Hal ini mempermudah dan mempercepat pengembangan sistem yang dikembangkan dengan metode analisa dan desain berorientasi obyek.

II.7. *NetBeans*

NetBeans merupakan salah satu *IDE* yang dikembangkan dengan bahasa pemrograman *java*. *NetBeans* mempunyai lingkup pemrograman java terintergrasi dalam suatu perangkat lunak yang di dalamnya menyediankan pembangunan pemrograman *GUI, text editor, complier,* dan *interpreter.NetBeans* adalah sebuah perangkat lunak *open source* sehingga dapat digunakan secara gratis untuk keperluan komersial maupun nonkomersial yang didukung oleh *Sun Microsystem*. (*Atik Rusmayanti ; 2013 : 2-3*)

II.8. *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan professional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama *Bubble chart, Bubble diagram*, model proses, diagram alur kerja, atau model fungsi.

DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

DFD ini merupakan alat perancangan system yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa

maupun rancangan ,sistem yang mudah dikomunikasikan oleh professional sistem kepada pemakai maupun pembuat program. (*Dahlan Abdullah ; 2013 : 154*)

II.9. UML (*Unified Modelling Language*)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, *Java*, C# atau *VB.NET*. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: *Grady Booch OOD* (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah:

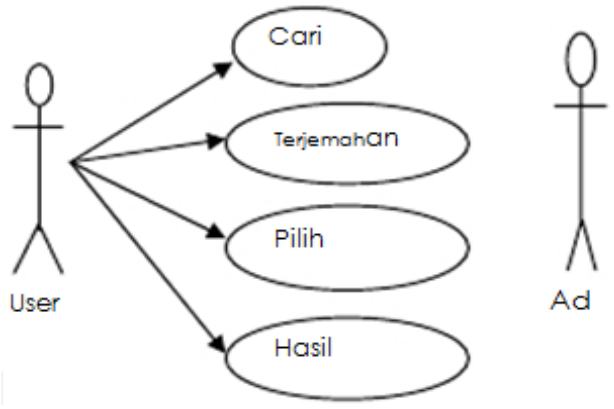
metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson,* yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. *(Yuni Sugiarti ; 2013 : 33)*

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterangkan dibawah ini.

1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor

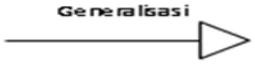
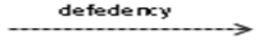
dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; 2013 : 41)*



Gambar II.2. *Use Case* Diagram
Sumber : (*Junaedi Siregar ; 2013 : 76*)

2. *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

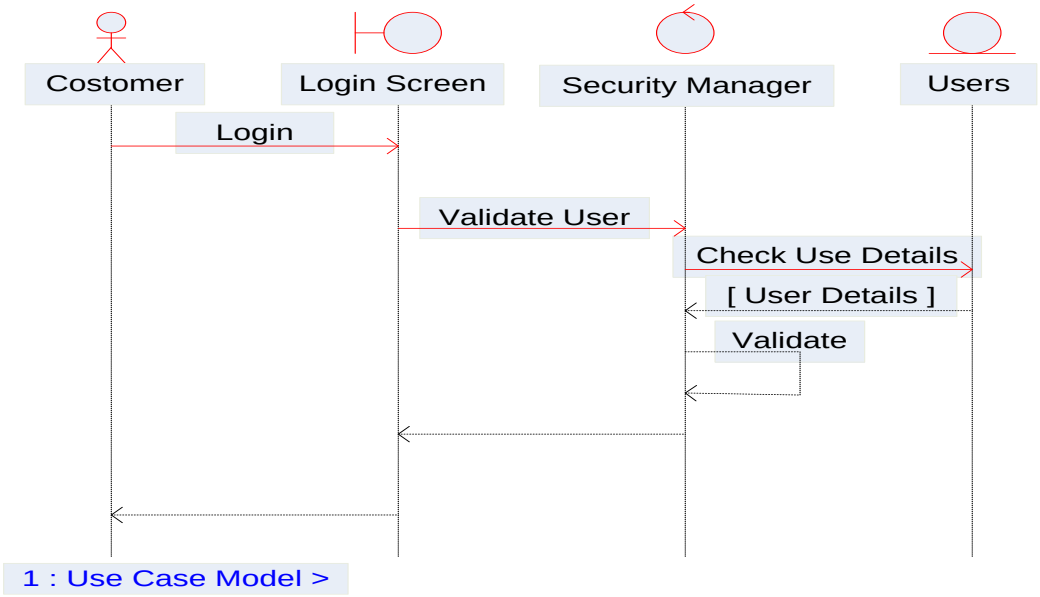
Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface 	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / defedency 	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	relasi antar kelas dengan makna semua-bagian (whole-part)

Gambar II.3. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.4. *Contoh Sequence Diagram*
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

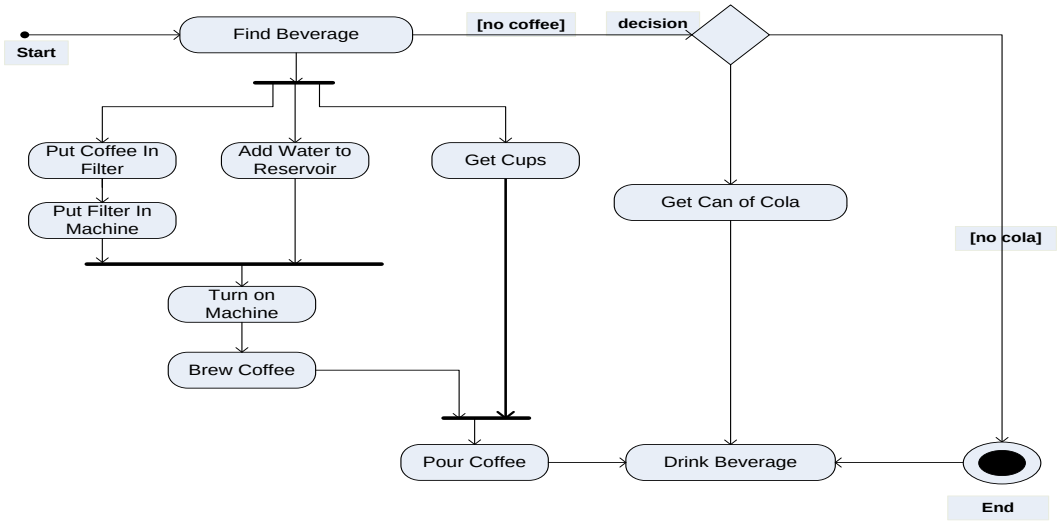
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.5. *Activify* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)