BAB 2
TINJAUAN PUSTAKA

2.1 Tinjauan Perusahaan

Tinjauan perusahaan menguraikan tentang perusahaan atau keterangan informasi dari tempat penelitian, mulai dari profil perusahaan, logo perusahaan, visi dan misi, struktur organisasi perusahaan, produk dan jasa yang ditawarkan, serta tugas dan wewenang.

2.1.1 Profil Perusahaan

PT. SKYLINE SEMESTA merupakan sebuah perusahaan swasta nasional yang pada awalnya bergerak dibidang IT (*Information Technology*) dengan memberikan layanan *Internet Service Provider* (ISP) untuk kalangan korporasi, pendidikan maupun *private customer.*

Namun atas permintaan *customer* maupun masyarakat, PT. SKYLINE SEMESTA pun dituntut untuk dapat memenuhi kebutuhan lainnya di bidang IT, baik itu *Security*, VPN, WAN, dan lain sebagainya dalam satu kesatuan. Dengan demikan akhirnya PT. SKYLINE SEMESTA dengan penuh keyakinan dan ditunjang oleh Sumber Daya Manusia dan Infrastruktur yang ada berkomitmen untuk menjadi salah satu perusahaan yang dapat menyediakan layanan total di bidang IT kepada setiap pelanggan.

2.1.2 Logo Perusahaan

Gambar 2.1 Logo PT. Skyline Semesta

2.1.3 Visi dan Misi

Menciptakan suatu inovasi produk dan jasa di bidang teknologi informasi dan komunikasi yang handal dan efektif. Dengan didukung pemberian pelayanan yang inovatif dan dinamis dengan tetap memenuhi kebutuhan dasar setiap pelanggan.

Memperkenalkan teknologi informasi kepada semua lapisan masyarakat serta membantu memberikan solusi yang tepat untuk setiap kebutuhannya. Hal ini dilakukan dengan memanfaatkan teknologi informasi secara efektif, efisien dan terjangkau dengan tetap memberikan kenyamanan dan keamanan menyeluruh terhadap teknologi yang digunakan.

2.1.4 Produk dan Jasa

Sebagai perusahaan yang berkomitmen untuk menjadi salah satu perusahaan yang dapat menyediakan layanan total di bidang IT kepada pelanggan, maka dengan sangat bangga kami paparkan produk dan jasa yang kami tawarkan.

2.1.4.1 Internet Service Provider

Skyline Semesta berusaha menjembatani kebutuhan masyarakat baik itu perusahaan/industri/institusi pendidikan maupun pemerintahan akan kebutuhan akses internet.

Didukung oleh jaringan yang solid, stabilitas *bandwidth* internet selalu terjaga dengan tingkat *availability* hingga 99%. Karenanya sudah menjadi komitmen kami yaitu untuk mampu memenuhi kebutuhan koneksi internet yang *reliable*, baik dari tingkat kestabilan koneksi yang terjaga, harga yang terjangkau dan dukungan teknis dan non teknis yang handal. Produk dan Layanan yang kami berikan adalah :

a. *Dedicated Connection*

   Ditujukan kepada pelanggan yang membutuhkan koneksi internet dengan penggunaan yang intensif dan optimal. Koneksi Skyline Semesta memiliki karakteristik *upstream* dan *downstream* yang

simetrik. Dengan demikian transfer data pada saat transfer (*upload*) maupun receive (*download*) akan mendapatkan kapasitas *bandwidth* yang optimal.

b. *Burstable Connection*

Ditujukan kepada pelanggan yang membutuhkan koneksi internet dengan penggunaan yang intensif tetapi dengan *budget* yang terjangkau.

2.1.4.2 Kaspersky Non Stop Security

Untuk menunjang dan memberikan proteksi penuh (*full protection*) pada sistem jaringan *LAN/WAN* dan *PC* (*Personal Computer*), Skyline Semesta menyediakan fasilitas program keamanan (*security*) pada *LAN/WAN* dan *PC* (*Personal Computer*) yaitu *Kaspersky Open Space Security*.

Khusus layanan *Kaspersky Open Space Security* yang kami kelola dan monitoring adalah *Kaspersky Non Stop Security*. Kelebihan dari *Kaspersky Open Space Security* adalah :

a. Memenuhi dan melampui semua kebutuhan proteksi dasar *network corporate*.
b. Sebuah solusi untuk semua titik dasar *network* dan *platform* (Microsoft, Linux, Mac OS, *Mobile Phones*).
c. Proteksi terhadap semua tipe ancaman internet dan *network.*
d. Respon sangat cepat terhadap *malware* baru.
e. Teknologi proaktif yang terintegrasi dengan proteksi tradisional *signature-based*.
f. *Kaspersky* menyediakan pengamanan disaat *mobile.*
g. Untuk menghalangi serangan *cyber crime* yang menyerang *network*, diproteksi oleh teknologi.
h. Teknologi *Anti – Roorkit, Phising, Hacker, Spyware* dan lain-lain.
i. Proteksi dari pencurian identitas dan data.
j. *Rollback* semua perubahan yang dibuat oleh *malicious software* (*virus, trojan, worm*, dll).

k. *Self-defense* terhadap serangan dari *malware.*

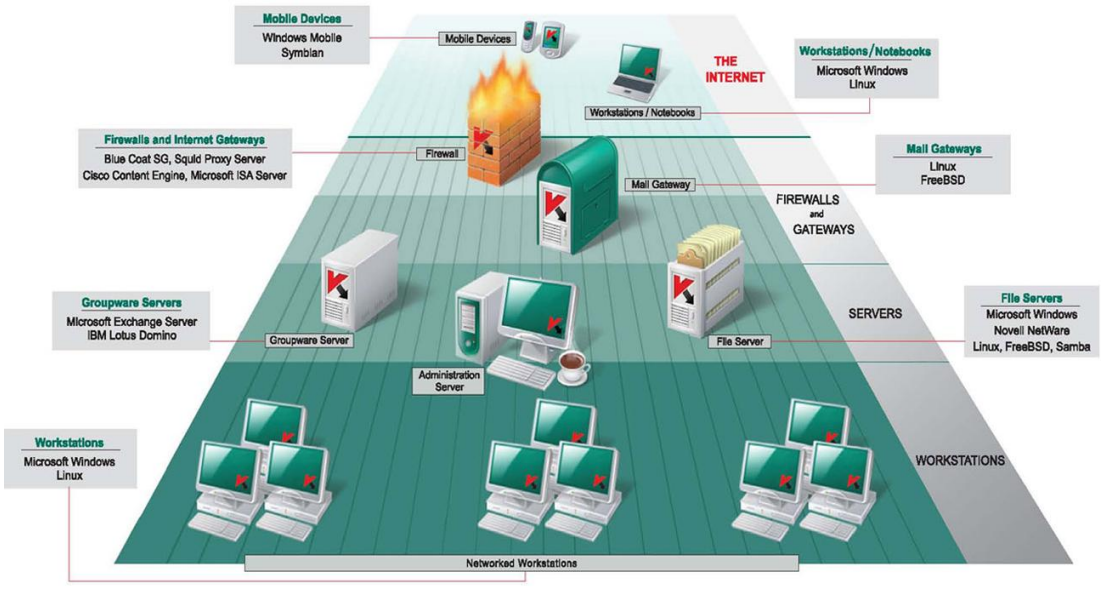
l. *Device control* dan *registry quad.*

m. *Kaspersky Lab Security* memudahkan Administrator bekerja secara transparan dan efisien:

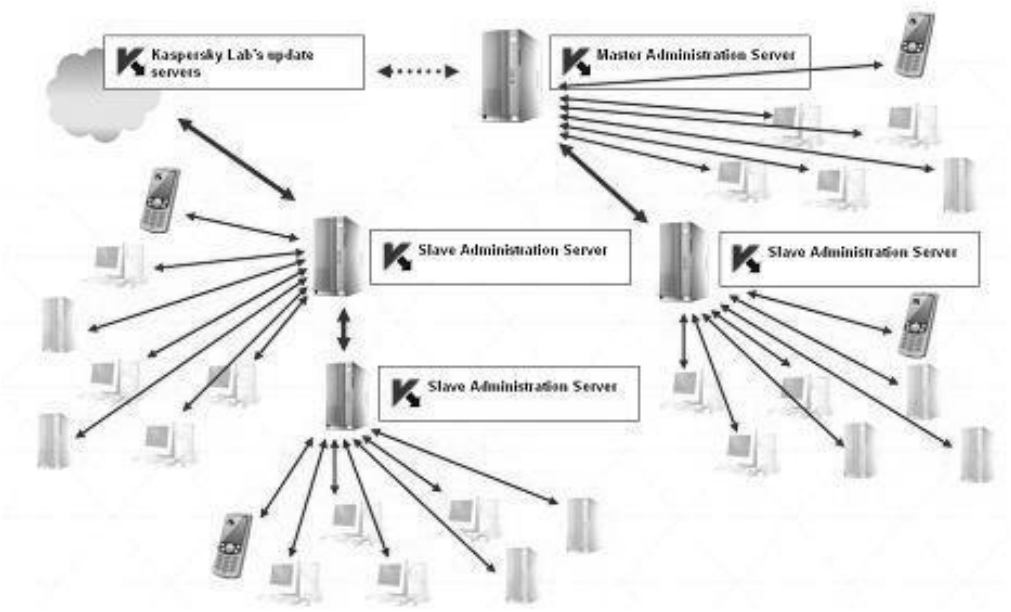
a) Administrasi terpusat.

b) Kompatibel dengan solusi lain.

c) Sangat efisien dalam menggunakan sumber daya *network* dan berbagai sarana akses internet *Bandwidth* yang kecil (CDMA, GPRS, ADSL, Modem Dial Up, Vsat dll)



Gambar 2.2 Sistem Monitoring Kaspersky Non Stop Security



Gambar 2.3 Diagram Jaringan Kaspersky Non Stop Security

Kaspersky Non Stop Security, mempermudah seorang admin jaringan dalam pengambilan keputusan apabila terjadi masalah dalam sebuah jaringan/*network.*

2.1.4.3 *Networking* dan *Server Administration*

Networking dan Administrasi *Server* adalah layanan yang berhubungan dengan pembangunan jaringan komputer lokal (*Local Area Network*), jaringan antar kantor hingga Jaringan antar pulau (*Wide Area Network*). Media koneksi dengan menggunakan kabel (*wire line*), *fiber optic* (FO) *wireless LAN* (tanpa kabel) dan *satelite network.*

Instalasi *hardware* dan *software* sesuai kebutuhan perusahaan atau institusi pelanggan semisal untuk Warung Internet, Laboratorium Komputer (SD, SMP, SMA/SMK) sampai Perguruan Tinggi ataupun perkantoran/bisnis dan Instansi Pemerintahan. Beberapa layanan dalam *networking* & *servers administration* adalah:

a. Instalasi dan Administrasi *Server*

Layanan Instalasi dan *Setting Server* merupakan solusi bagi perusahaan dan institusi lainnya untuk pembagunan *data center* sebagai pusat kendali sistem yang handal. Sangat diperlukan untuk implementasi Perusahaan Lokal, Perusahaan Asing, Instansi Pemerintahan, Universitas, dan lembaga lain yang sudah menerapkan Sistem Informasi Manajemen Elektronik. Adapun Implementasi *Server* tersebut antara lain :

a) *Windows Base Server*

Windows Server 2003, Domain Controller, Active Directory, Internet Information System, Router and Remote Access, DHCP Service, Microsoft Exchange Server, Microsoft SQL Server, Oracle DataBase Server.

b) *Linux Enterprise Base Server*

DNS Server, Web Server, Mail Server, Proxy Server, File Server, Database Server (MySQL, PostgreSQL, Oracle), Application Server, DHCP Server, Router & Firewall.

b. *Local Area Network* (LAN)

Layanan pemasangan jaringan mulai dari jaringan lokal (LAN) untuk menunjang komunikasi data antar komputer pada perusahaan atau syarat terjadinya integrasi Sistem Informasi Manajemen.

Layanan ini dapat disesuaikan dengan kebutuhan perusahaan dengan jenis-jenis layanan kami sebagai berikut :

a) Installasi Kabel Jaringan Lokal dalam satu ruangan, gedung atau perkantoran yang bisa dijangkau dengan kabel.
b) Installasi dan konfigurasi komputer *client* agar dapat terhubung ke *Central* (Pusat Data Elektronik).
c) Penyediaan *Hardware* dan *Software* pendukung Jaringan.

c. Wireless Network

Wireless Network merupakan solusi jaringan antar gedung untuk menghubungkan antar jaringan LAN & PC yang lokasinya berjauhan dalam satu wilayah (*Metropolitan Area Network*) dengan

menggunakan teknologi *Broadband Radio Link* dengan kecepatan 2 - 36 Mbps. Implementasi jaringan *Wireless* biasanya juga digunakan untuk berbagi akses internet agar memperoleh akses internet secara online 24 jam dengan biaya yang murah.

d. *Hot Spot Installation*

Jasa instalasi layanan instalasi *Hot Spot* dengan peralatan yang sederhana ataupun canggih.

e. *Maintenance Network*

Layanan ini ditujukan bagi perusahaan/organisasi skala kecil dan menengah ataupun perusahaan besar yang tidak memiliki Divisi Teknologi Informasi sendiri dapat mempercayakan IT *Outsourcing*-nya kepada kami. Layanan ini membantu perusahaan agar seakan-akan perusahaan tersebut memiliki Bagian/Divisi Teknologi Informasi-nya sendiri. Setelah Sistem dan Infrastruktur IT-nya terbangun dan beroperasi dengan baik, kami berikan *support* secara kontinyu untuk menjamin operasionalisasi dan keamanan sistem antara lain:

a) Memantau Sekuritas Sistem.
b) Penanganan Anti-virus dan anti-spam.
c) Manajemen konektivitas Internet.
d) Memantau Operasional *Server.*
e) Manajemen *Electronic-mail* system.
f) Manajemen *bandwidth* dan monitoring.
g) Manajemen *Router, Proxy* dan *Firewall*

2.1.4.4 Komunikasi Voice dan Video

a. Instalasi *VoIP (Voice over IP)*

Layanan instalasi *VoIP* dengan menggunakan jaringan *VoIP public* seperti Skype, *VoIP* Rakyat dan lain - lain

b. Instalasi *Video Conference*

Layanan instalasi *Video Conference* untuk keperluan konferensi, rapat atau presentasi jarak jauh dengan menggunakan perangkat *hardware*

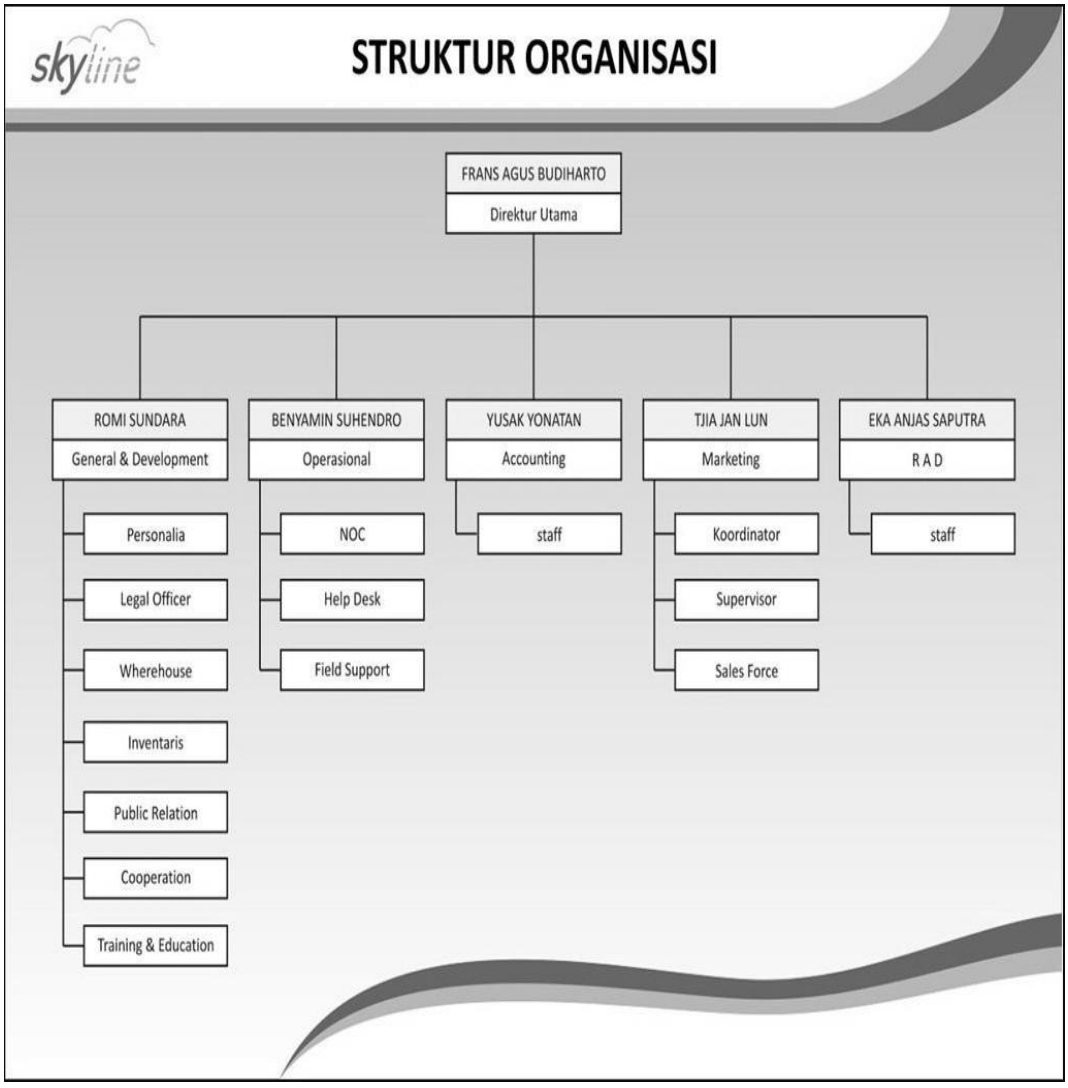
serta *software* berkualitas dalam rangka meningkatkan efisiensi waktu dan biaya.

2.1.4.5 Pengadaan *Hardware* dan *Software*

Kami adalah master *dealer hardware/software* dan distributor beberapa perangkat keras komputer serta menjalin kerjasama dengan produsen dan distributor resmi perangkat komputer, *peripheral*, *accessories* dan *software*.

2.1.5 Struktur Organisasi Perusahaan

Berikut merupakan Struktur Organisasi PT. Skyline Semesta.



Gambar 2.4 Struktur Organisasi PT. Skyline Semesta

2.1.6 *Job Description*

Deskripsi tugas merupakan rincian dari fungsi, tugas, wewenang, tanggung jawab, dan hubungan kerja dari setiap personil yang menduduki jabatan tertentu. PT Skyline Semesta mempunyai struktur organisasi yang ditetapkan berdasarkan kebijakan dari direktur. Tugas, wewenang, dan tanggung jawab masing-masing bagian adalah sebagai berikut:

a. Direktur Umum
    1. Memimpin perusahaan secara keseluruhan.
    2. Menentukan visi dan misi perusahaan.
    3. Menentukan rencana kerja agar visi dan misi tercapai.

b. *General & Development*
    1. Bertanggung jawab terhadap urusan kepegawaian, inventaris, dan kerja sama dengan perusahaan lain.
    2. Mengurus perangkat dan peralatan teknik.
    3. Membuat laporan kegiatan bagian umum.

c. Operasional
    1. Mengendalikan sistem dan cara kerja di unitnya agar menciptakan suasana kondusif.
    2. Mengkoordinasikan dan mengarahkan penyusunan rencana dan target kerja.
    3. Mengkoordinasikan unitnya agar mencapai target dari rencana kerja.

d. *Accounting*
    1. Membuat tagihan dan surat tagihan untuk pelanggan.
    2. Menginput penerimaan pembayaran dari pelanggan dan pembayaran ke suplier.
    3. Mengarsip seluruh dokumen transaksi.

e. Marketing
    1. Membuat strategi dan segmentasi pasar untuk penjualan perusahaan.
    2. Merencanakan program-program pemasaran.

3. Mengkoordinir pencarian informasi pasar.

f. *R&D (Research And Development)*
   1. Pengembangan sistem.
   2. Melakukan penelitian terhadap teknologi baru.
   3. Melakukan implementasi sistem baru.

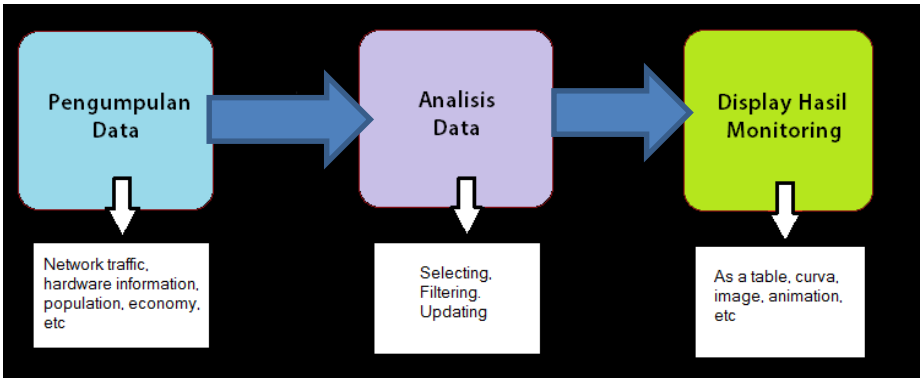
2.2 Landasan Teori

Landasan teori berisi tentang teori-teori yang dipakai dan mendukung dalam kegiatan penelitian untuk membangun aplikasi monitoring jaringan berbasis android.

2.2.1 Sistem Monitoring

Sistem monitoring merupakan suatu proses untuk mengumpulkan data dari berbagai sumber daya. Biasanya data yang dikumpulkan merupakan data yang *real time*. Secara garis besar tahapan dalam sebuah sistem monitoring terbagi ke dalam tiga proses besar seperti yang terlihat pada gambar 2.5 , yaitu:

1. Proses di dalam pengumpulan data monitoring.
2. Proses di dalam analisis data monitoring.
3. Proses di dalam menampilkan data hasil monitoring.



Gambar 2.5 Proses Sistem Monitoring

Aksi yang terjadi di antara proses-proses dalam sebuah sistem monitoring adalah berbentuk *service*, yaitu suatu proses yang terus-menerus berjalan

pada interval waktu tertentu. Proses-proses yang terjadi pada suatu sistem monitoring dimulai dari pengumpulan data seperti data dari *network traffic*, *hardware information*, dan lain-lain yang kemudian data tersebut dianalisis pada proses analisis data dan pada akhirnya data tersebut akan ditampilkan.

Pada beberapa aplikasi sistem monitoring, akses benar-benar dibatasi dari *local host* terminal saja. Pertanyaannya apakah bisa dilakukan monitoring dari jarak jauh, dimana semua data yang dikumpulkan dari terminal komputer yang berada di lokasi berbeda dengan instrumennya misalnya dengan menggunakan jaringan LAN (*Local Area Network*) atau bahkan internet. Untuk menjalankan sistem monitoring yang seperti ini sangat memungkinkan sekali dapat dilakukan dengan menggunakan *interface* program yang dapat menjembatani pengguna melalui *web browser* pada *remote terminal*. *Interface* program ini disebut CGI (*Common Gateway Interface*) yang biasanya tersedia pada linux [3].

2.2.2 Jaringan Komputer

Menurut Ardiansyah (2010), jaringan komputer (*Computer Network*) yang disebut secara singkat dengan jaringan adalah kumpulan komputer dan alat alat yang lain yang saling dihubungkan bersama menggunakan media komunikasi tertentu. Informasi yang melintas sepanjang media komunikasi, memungkinkan pengguna jaringan untuk saling bertukar data atau menggunakan perangkat lunak maupun perangkat keras secara berbagi. Masing - masing komputer yang atau alat - alat lain yang di hubungkan pada jaringan disebut *node*. Jaringan dapat terdiri dari puluhan, ratusan atau bahkan ribuan *node*.

2.2.2.1 *IP Address*

IP address digunakan untuk mengidentifikasi *interface* jaringan pada *host* dari suatu komputer. Dengan adanya *IP address* masing-masing host dapat terhubung dan saling bertukar informasi melalaui media transmisi kabel seperti UTP, koaksial atau *fiber optic*. Sebagai contoh sederhana, jika sebuah surat akan dikirimkan/ditujukan ke orang lain maka surat tersebut harus dilengkapi dengan alamat lengkap si penerima. Tentu juga alamat si pengirim perlu dicantumkan

untuk memudahkan penerima dari mana datangnya surat tersebut. Jika alamat si penerima tidak lengkap misalnya tidak ada nomor rumah, tidak di cantumkan nama penerima maka surat tersebut dipastikan tidak akan sampai.

IP address adalah sekelompok bilangan biner 32 bit yang dibagi menjadi 4 bagian yang masing-masing bagian itu terdiri dari 8 bit, angka pada masing-masing bit tersebut adalah angka 1 dan 0. misalnya : 11000111. Nilai paling besar dari biner 8 bit adalah 255, angka 255 ini dihitung dari bilangan biner 2 berpangkat.

Misalnya :

11111111 = 27 + 26 + 25 + 24 + 23 + 22 + 21 + 20

= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1

= 255

Dengan demikian *IP address* yang terdiri dari 4 bagian bilangan 8 bit maka nilai terbesar *IP address* tersebut adalah

11111111.11111111.11111111.11111111 atau 255.255.255.255.255

Untuk memudahkan kita dalam membaca dan mengingat suatu alamat IP maka umumnya penamaan yang digunakan adalah berdasarkan bilangan desimal. *IP address* dibagi menjadi kelas-kelas yang masing-masing mempunyai kapasitas jumlah IP yang berbeda-beda. *IP address* terdiri dari dua bagian yaitu bagian *network ID* dan *host ID*. *Network ID* menunjukkan ID alamat jaringan tempat *host-host* berada sedangkan *host ID* adalah bagian yang menunjukkan *host* itu berada. Sederhananya, *Network ID* seperti nama jalan sedangkan *Host ID* adalah nomor rumah di jalan tersebut [4].

2.2.2.2 Kelas *IP Address*

IP address harus dikelompokan dalam kelas-kelas. Dasar pertimbangan pembagian *IP address* ke dalam kelas-kelas adalah untuk mempermudah pendistribusian pendaftaran *IP address*. Secara umum *IP address* yang sering dipakai dikelompokan dalam tiga kelas yaitu [4]:

1. Kelas A

IP address kelas A terdiri dari 8 bit untuk *network ID* dan sisanya 24 bit digunakan untuk *host ID*, sehingga *IP address* kelas A digunakan untuk jaringan dengan jumlah host sangat besar. Pada bit pertama berikan angka 0 sampai dengan 127.

2. Kelas B

   IP address kelas B terdiri dari 16 bit untuk network ID dan sisanya 16 bit digunakan untuk *host ID*, sehingga *IP address* kelas B digunakan untuk jaringan dengan jumlah *host* tidak terlalu besar. Pada 2 bit pertama berikan angka 10 sehingga bit awal IP tersebut mulai dari 128 – 191.

3. Kelas C

   IP address kelas C terdiri dari 24 bit untuk *network ID* dan sisanya 8 bit digunakan untuk *host ID*, sehingga *IP address* kelas C digunakan untuk jaringan untuk ukuran kecil. Kelas C biasanya digunakan untuk jaringan *Local Area Network* atau LAN. Pada 3 bit pertama berikan angka 110 sehingga bit awal IP tersebut mulai dari 192 – 223.

2.2.3 *Simple Network Managemen Protocol*(SNMP)

SNMP merupakan suatu sistem manajemen yang vital untuk memantau dan mengumpulkan informasi tentang "kesehatan" suatu jaringan. Dengan pemantauan dan pengumpulan informasi tersebut dapat dilakukan analisis. Salah satu metode analisis adalah analisis jangka pendek (*short-term analysis*) yang berguna untuk mendeteksi dan membuat perbaikan secara cepat atas kesalahan dan masalah yang timbul [5]. Dengan demikian jaringan akan memberikan *Quality of Service* (QoS) yang lebih baik

SNMP juga merupakan protokol yang dirancang agar mampu memberikan layanan manajemen jaringan yang handal kepada pengguna, sehingga pengguna dapat memantau dan memelihara jaringan komputernya secara sistematis dari jarak jauh atau pada suatu pusat pemantauan. Sistem pengelolaan yang dijalankan adalah mengumpulkan data dan melakukan penetapan terhadap variabel-variabel dalam elemen jaringan yang dikelola [6].

2.2.3.1 Protokol SNMP

Menurut standar IETF, SNMP didesain untuk pemakaian di Internet yang didesain diatas protokol UDP (*User Diagram Protocol*) seperti yang digambarkan pada Tabel 2.1.

Tabel 2.1 Lapisan Pada Protokol SNMP

Sistem Manajemen Jaringan
Agent
SNMP
UDP
IP
Lapisan Bawah

Karena menggunakan protokol UDP, SNMP adalah protokol yang *connectionless.* Tidak ada jaminan lalu lintas manajemen diterima oleh entitas lain dengan sempurna. Dengan protokol ini, *overload* proses dapat dikurangin dan diperoleh kesederhanaan.

Cara kerja SNMP adalah dengan jalan Manajer dan *agent* saling berkirim pesan berupa permintaan manager dan jawaban dari agent tentang informasi jaringan yang dibawa oleh paket-paket data yang disebut PDU (*Protocol Data Unit*).

2.2.4 *Network Monitoring System*(NMS)

NMS merupakan *tools* untuk melakukan monitoring/pengawasan pada elemen-elemen dalam jaringan komputer. Fungsi dari NMS adalah melakukan pemantauan terhadap kualitas SLA (*Service Level Agreement*) dari *bandwidth* yang digunakan (Fachruddin, 2009). Hasil dari pantauan tersebut biasanya dijadikan bahan dalam pengambilan keputusan oleh pihak manajemen, disisi lain digunakan oleh Administrator jaringan (*technical person*) untuk menganalisa apakah terdapat kejanggalan dalam operasional jaringan.

Menurut Ipswitch (2010), ada 10 alasan utama menggunakan aplikasi monitoring jaringan komputer, yaitu:

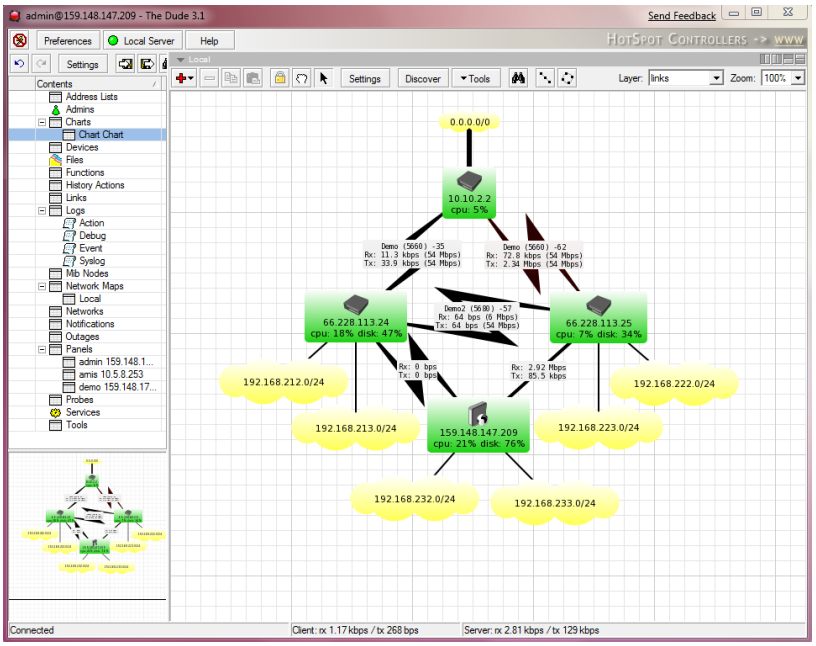
a. Mengetahui apa yang sedang terjadi dalam jaringan, dimana solusi NMS selalu memberikan informasi tentang operasional dan konektivitas dari peralatan dan sumber daya yang ada dalam jaringan.
b. Untuk perencanaan peningkatan (*upgrade*) dan perubahan peralatan jaringan.
c. Dapat digunakan untuk mendiagnosa masalah-masalah dalam jaringan.
d. Sebagai bahan untuk keperluan SLA (*service level agreement*).
e. Mengetahui kapan saat yang tepat untuk mengimplementasikan solusi *disaster recovery system* (pemulihan bencana/masalah) dapat dilaksanakan.
f. Memastikan keamanan sistem beroperasi dengan baik.
g. Memastikan pengguna (*client*) layanan dalam jaringan terkoneksi dengan *server* yang mereka butuhkan.
h. Mendapatkan infomasi status jaringan secara *remote.*
i. Memastikan *uptime* untuk keperluan pengguna yang tergantung dengan ketersediaan jaringan komputer.
j. Menghemat pengeluaran dengan menekan jumlah waktu jaringan *down* dan memangkas waktu untuk menganalisa masalah.

2.2.4.1 Aplikasi NMS

Terdapat beberapa aplikasi *Network Managemen System*(NMS) yang digunakan dalam pengerjaan penelitian ini adalah sebagai berikut :

a. The Dude

The Dude Network Monitor merupakan aplikasi MikroTik yang dapat secara signifikan meningkatkan jangkauan Administrator dalam mengelola lingkungan jaringan. The Dude akan otomatis memindai semua perangkat di dalam subnet tertentu, menggambar dan membuat tata letak peta jaringan, memberikan layanan memonitor perangkat-perangkat yang terhubung dalam jaringan dan memberikan peringatan jika suatu layanan jaringan mengalami masalah. Bentuk monitoring NMS The Dude dapat dilihat pada gambar 2.6.



Gambar 2.6 Tampilan NMS The Dude

b. OpenNMS

OpenNMS ialah sebuah alat manajemen jaringan berbasis *open source* skala *enterprise*. OpenNMS membantu seorang administrator jaringan untuk memantau layanan yang terganggu pada peralatan atau perangkat di sisi *remote* (jauh) dan mengumpulkan formasi dari simpul perangkat *remote* tersebut dengan menggunakan SNMP. OpenNMS menyediakan. OpenNMS ditulis dalam bahasa JAVA namun, konfigurasi data tersedia secara langsung lewat *data extensible markup language* (XML). Satu hal yang cukup penting ialah OpenNMS mampu bekerja pada sistem bertingkat (hierarkikal) dan mampu untuk memonitoring beberapa *service*/layanan seperti : ICMP, SNMP, FTP, HTTP, SMTP, DNS, Router TCP, Sybase (TCP), MySQL, Postgres, Oracle (TCP), DHCP, MS Exchange, IMAP, POP3 dan lain-lain. OpenNMS juga menawarkan jasa, dukungan, serta pengembangan dan pelatihan untuk membuat solusi *custom* yang dibuat untuk menyesuaikan proses bisnis [7].

Adapun fitur-fitur yang terdapat pada openNMS antara lain:

1. *Service polling*

   Menentukan ketersediaan layanan dan latensi, termasuk pengukuran didistribusikan ketersediaan dan latensi, dan pelaporan hasil.

2. Pengumpulan data

   Mengumpulkan, menyimpan dan melaporkan data yang dikumpulkan dari *node* melalui protokol termasuk SNMP, JMX, HTTP, Windows Management Instrumentation, dan NSClient.

3. *Thresholding*

   Mengevaluasi data *latency* yang disurvei atau data kinerja yang dikumpulkan.

4. *Event* manajemen

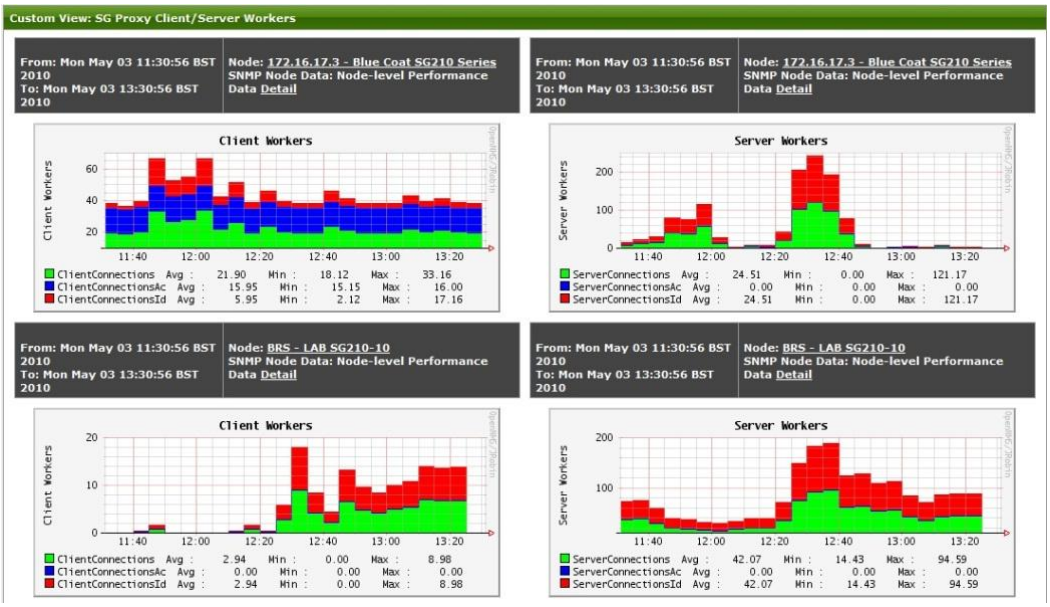
   Menerima peristiwa, baik internal maupun eksternal, termasuk melalui perangkap SNMP

5. Alarm dan otomatisasi

   Mengurangi peristiwa menurut kunci pengurangan dan *scripting* tindakan otomatis berpusat di sekitar alarm.

6. Pemberitahuan

   Pemberitahuan tentang kejadian penting mengirimkan melalui e-mail, XMPP, atau cara lain.



Gambar 2.7 Tampilan NMS OpenNMS

2.2.5 Android

Android adalah kumpulan perangkat lunak yang ditujukan bagi perangkat bergerak mencakup sistem operasi, *middleware*, dan aplikasi kunci. Android *Standart Development Kid* (SDK) menyediakan perlengkapan dan *Application Programming Interface* (API) yang diperlukan untuk mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java.

Android dikembangkan oleh Google bersama *Open Handset Allience* (OHA) yaitu aliansi perangkat selular terbuka yang terdiri dari 47 perusahaan *Hardware, Software* dan perusahaan telekomunikasi ditujukan untuk mengembangkan standar terbuka bagi perangkat selular.

Android dianggap sebagai platform masa depan yang lengkap, terbuka dan bebas sebagai berikut [8] :

a. Lengkap (*Complete Platform*)

Para desainer dapat melakukan pendekatan yang komprehensif ketika mereka sedang mengembangkan platform Android. Android merupakan sistem operasi yang aman dan banyak menyediakan *tools* dalam II-17 membangun *software* dan memungkinkan untuk peluang pengembangan aplikasi.

b. Terbuka (*Open Source*)

Platform Android disediakan melalui lisensi *open source*. Pengembang dapat dengan bebas untuk mengembangkan aplikasi. Android sendiri menggunakan Linux kernel 2.6.

c. Bebas (*Free Platform*)

Android adalah platform atau aplikasi yang bebas *develop*. Tidak ada lisensi atau biaya royalti untuk dikembangkan pada *platform* Android. Tidak ada biaya keanggotaan diperlukan. Tidak diperlukan biaya pengujian. Tidak ada kontrak yang diperlukan. Aplikasi untuk android dapat didistribusikan dan diperdagangkan dalam bentuk apapun.

2.2.5.1 Sejarah Android

Pada mulanya terdapat berbagai macam sistem operasi pada perangkat seluler, diantaranya sistem operasi Symbian, Microsoft Windows Mobile, Mobile Linux, iPhone, dan sistem operasi lainnya. Namun diantara sistem operasi yang ada belum mendukung standar dan penerbitan API yang dapat dimanfaatkan secara keseluruhan dan dengan biaya yang murah. Kemudian Google ikut berkecimpung didalamnya dengan platform Android, yang menjanjikan keterbukaan, keterjangkauan, *open source*, dan *framework* berkualitas.

Pada tahun 2005, Google mengakuisisi perusahaan Android Inc. untuk memulai pengembangan *platform* android. Dimana terlibat dalam pengembangan ini Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada pertengahan 2007 sekelompok pemimpin industri bersama-sama membentuk aliansi perangkat selular terbuka, Open Handset Alliance (OHA). Bagian dari tujuan aliansi ini adalah berinovasi dengan cepat dan menanggapi kebutuhan konsumen dengan lebih baik, dengan produk awalnya adalah platform Android. Dimana Android dirancang untuk melayani kebutuhan operator telekomunikasi, manufaktur *handset*, dan pengembang aplikasi. OHA berkomitmen untuk membuat android *open source* dengan lisensi Apache versi 2.0.

Android pertama kali diluncurkan pada 5 November 2007, dan *smartphone* pertama yang menggunakan sistem operasi android dikeluarkan oleh T-Mobile dengan sebutan G1 pada bulan September 2008 [9].

2.2.5.2 Perkembangan Android

Seiring dengan pengembangannya, Android memiliki berbagai macam versi, antara lain :

a. Android versi 1.1

   Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan *email*.

b. Android versi 1.5 (*Cupcake*)

   Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (*Software Development Kit*) dengan versi 1.5 (*Cupcake*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke *headset Bluetooth*, animasi layar, dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

c. Android versi 2.0 – 2.1 (*Eclair*)

   Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (*Eclair*), perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru.

d. Android versi 2.2 (*Froyo*)

   Pada 20 Mei 2010, Android versi 2.2 (*Froyo*) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5

kali lebih cepat, intergrasi V8 JavaScript *engine* yang dipakai Google Chrome yang mempercepat kemampuan *rendering* pada *browser*, pemasangan aplikasi dalam *SD Card*, kemampuan WiFi *Hotspot* portabel, dan kemampuan *auto update* dalam aplikasi Android Market.

e. Android versi 2.3 – 2.3.7 (*Gingerbread*)

Versi ini dirilis pada 6 Desember 2010. Gingerbread adalah kue yang terbuat dari jahe. *Gingerbread* dipakai sebagai nama alias dari sistem operasi Android versi 2.3. Ada banyak peningkatan pada versi Android yang satu ini dibandingkan dengan versi sebelumnya, seperti memaksimalkan kemampuan aplikasi dan game, dukungan layar resolusi WXGA dan diatasnya, serta mulai digunakannya *Near Field Communication* (NFC). Beberapa versi *update* yang dirilis antara lain v.2.3.3 hingga v.2.3.7.

f. Android versi 3.0 -3.2 (*Honeycomb*)

Versi ini dirilis pada 22 Februari 2011. *Honeycomb* atau sarang madu, dipakai sebagai nama alias dari sistem Android versi 3.0 Android versi ini merupakan OS yang didesain khusus untuk pengoptimalan pengunaan pada tablet PC.

g. Android versi 4.0 – 4.0.4 (*Ice Cream Sandwich*)

Versi ini dirilis pada 19 Oktober 2011. *Ice Cream Sandwich* dipakai sebagai nama alias dari Android versi 4.0. Secara teori semua perangkat seluler yang menggunakan versi Android sebelumnya, *Gingerbread*, dapat di-*update* ke Android *Ice Cream Sandwich*.

h. Android versi 4.1 – 4-3 (*Jelly Bean*)

Android *Jelly Bean* memiliki penambahan baru diantaranya meningkatkan *input keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Terdapat pula aplikasi Google Now yang dapat memberikan informasi yang tepat pada waktu yang tepat pula. Selain itu dapat mengetahui informasi cuaca, lalu-lintas, ataupun hasil pertandingan olahraga. Sistem operasi Android *Jelly Bean* 4.1 kini sudah mencapai versi *Jelly Bean* 4.3.

i. Android versi 4.4 (*KitKat*)

Android *KitKat* diluncurkan pada Oktober 2013. Penamaan pada versi ini didasarkan dari kerja sama antara Google dengan perusahaan Nestle. Android versi 4.4 ini diprediksikan akan kompatibel untuk digunakan pada smartphone Android mulai dari kelas *high-end* hingga *low-end*.

j. Android vesi 5.0 (*Lollipop*)

Android *Lollipop* pertama kali dirilis pada 25 Juni 2014, sebelumnya sistem operasi ini diperkenalkan dengan kode Android L. *Update* penggunaan sistem operasi Android dari versi sebelumnya ke versi 5.0 pertama kali dibuka pada 12 November 2014. *Interface* baru digunakan serta beberapa perbaikan pada akses notifikasi yang lebih mudah. Perubahan *platform* dengan Android *Runtime* menggantikan Dalvik dilakukan guna meningkatkan performa aplikasi dan juga mengoptimalkan konsumsi energi dari baterai.

2.2.5.3 Fitur Android

Adapun beberapa fitur – fitur yang tersedia di Android adalah sebagai berikut ini [8]:

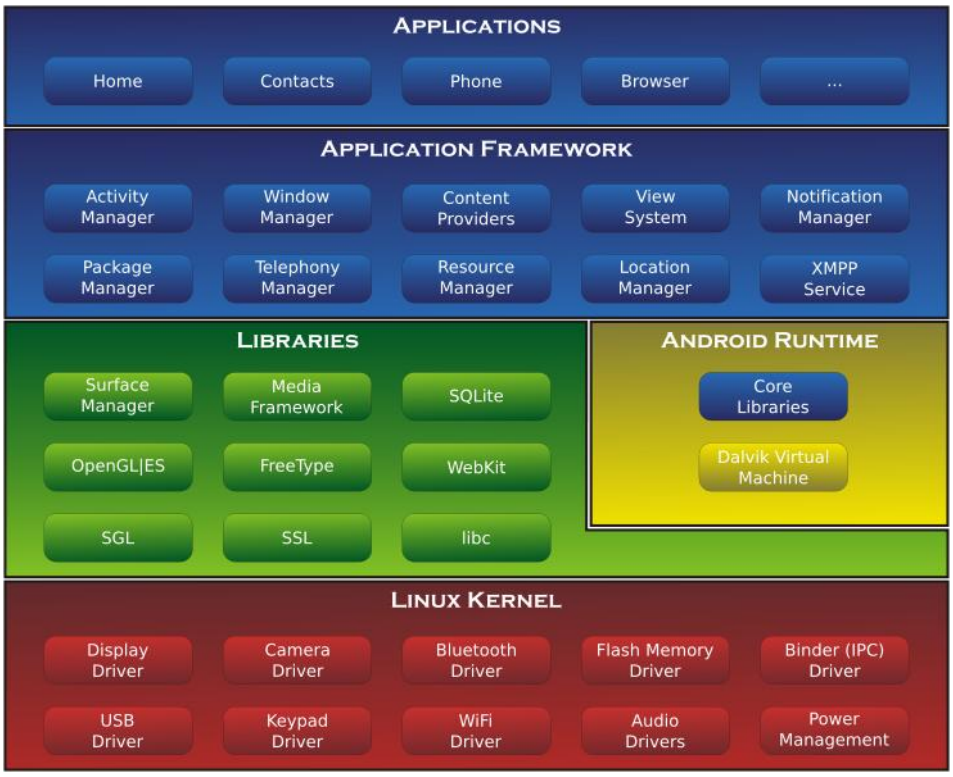
- Kerangka aplikasi: memungkinkan penggunaan dan penghapusan komponen yang tersedia.
- Dalvik mesin virtual: mesin virtual dioptimalkan untuk perangkat telepon seluler.
- Grafik: grafik di 2D dan grafis 3D berdasarkan pustaka OpenGL.
- SQLite: untuk penyimpanan data.
- Mendukung media: audio, video, dan berbagai format gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- GSM, Bluetooth, EDGE, 3G, 4G dan WiFi (tergantung piranti keras)
- Kamera, *Global Positioning System* (GPS), kompas, NFC dan *accelerometer* (tergantung piranti keras).

2.2.5.4 Android SDK (*Software Development Kit*)

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mengembangkan aplikasi pada platform Android yang menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Sebagai platform Android aplikasi – netral, Android memberi Anda kesempatan untuk membuat Aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone/Smartphone* [8].

2.2.5.5 Arsitektur Android

Arsitektur Android merupakan sebuah *kernel* Linux dan sekumpulan pustaka C / C++ dalam suatu *framework* yang menyediakan dan mengatur alur proses. Berikut adalah gambar arsitektur android [8]:



Gambar 2.8 Arsitektur Android [8]

- *Application and Widget*

  Applications and Widget adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Puncak dari diagram arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan *service* yang tersedia pada *framework* aplikasi. Lapisan aplikasi Android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada Android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.
- *Application Frameworks*

  Applications Frameworks adalah *layer* dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi android. Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Selain itu, juga menyediakan abstraksi *generic* untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi.
- *Libraries*

  Libraries adalah *layer* dimana fitur-fitur android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

  1) *Media Library* untuk memutar dan merekam berbagai macam format audio dan video.

2) *Surface Manager* untuk mengatur hak akses *layer* dari berbagai aplikasi.
3) *Graphic Library* termasuk didalamnya SGL dan OpenGL, untuk tampilan 2D dan 3D.
4) *SQLite* untuk mengatur relasi *database* yang digunakan pada aplikasi.
5) SSl dan *WebKit* untuk *browser* dan keamanan internet. Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

- *Android Run Time*

  Pada Android tertanam paket pustaka inti yang menyediakan sebagian besar fungsi Android. Inilah yang membedakan Android dibandingkan dengan sistem operasi lain yang juga mengimplementasikan Linux. Android *Runtime* merupakan mesin virtual yang membuat aplikasi Android menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam Android Runtime terdapat 2 bagian utama, diantaranya:

  1) Pustaka Inti, Android dikembangkan melalui bahasa pemrograman Java, tapi Android *Runtime* bukanlah mesin virtual Java. Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.
  2) Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format Dalvik Executable (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan diatas kernel Linux 2.6,

dengan fungsi dasar seperti *threading* dan manajemen memori yang terbatas.

- *Linux Kernel*

  Android dibangun di atas *kernel* Linux 2.6. Namun secara keseluruhan Android bukanlah linux, karena dalam Android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada Android hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan *driver*. Kernel linux menyediakan driver layar, kamera, keypad, WiFi, Flash Memory, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan.

2.2.5.6 Komponen Aplikasi

Fitur penting android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada.

Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

2.2.5.6.1 Activities

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *Activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *Activity* dideklarasikan dalam sebuah kelas yang bertugas untuk

menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity* atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan.

Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack.* Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan *acitivity*, LIFO (*Last In First Out*) dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol *Back*, atau menutup *activity* yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. *Memory Manager* android menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan *activity*, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut.

Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi [10]:

1) **Active,** setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus, dan menerima masukkan dari pengguna. Android akan berusaha untuk membuat *activity* aplikasi ini untuk untuk tetap hidup dengan segala cara, bahkan akan menghentikan *activity* yang berada dibawah tumpukkannya jika diperlukan. Ketika *activity* sedang aktif, maka yang lainnya akan dihentikan sementara.
2) **Paused,** dalam beberapa kasus *activity* akan terlihat tapi tidak terfokus pada kondisi inilah disebut *paused.* Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan *paused*, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna. Dalam kasus ekstrim, android akan menghentikan *activity* dalam keadaan *paused* ini, untuk menunjang sumber daya bagi *activity* yang sedang aktif.
3) **Stopped,** ketika sebuah *activity* tidak terlihat, maka itulah yang disebut *stopped. Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan menjadi kandidat utama untuk

dieksekusi oleh sistem ketika membutuhkan sumberdaya lebih. Oleh karenanya ketika suatu *activity* dalam kondisi *stopped* maka perlu disimpan data dan kondisi antarmuka saat itu. Karena ketika *activity* telah keluar atau ditutup, maka dia akan menjadi *inactive.*

4) **Inactive,** kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* telah ditiadakan dari tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali.

Kondisi transisi ini sepenuhnya ditangani oleh manajer memori android. Android akan memulai menutup aplikasi yang mengandung *activity inactive*, kemudian *stopped activity*, dan dalam kasus luar biasa *paused activity* juga akan di tutup.

2.2.5.6.2 *Services*

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *Activity* tidak aktif atau tidak tampak.

2.2.5.6.3 *Intents*

Intens merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman web, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misal pesan masuk. *Intents* dapat eksplisit atau implisit, contohnya jika suatu aplikasi ingin menampilkan *URL*, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

2.2.5.6.4 *Broadcast Receivers*

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan

pemberitahuan. Sebagian besar *Broadcast* berasal dari sistem misalnya, Batre sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa *default* pada perangkat. Sama halnya dengan *service, Broadcast Receivers* tidak menampilkan antarmuka pengguna. Namun, *Broadcast Receivers* dapat menggunakan *Notification Manager* untuk memberitahukan sesuatu kepada pengguna.

2.2.5.6.5 *Content Providers*

Content Providers digunakan untuk mengelola dan berbagi database. Data dapat disimpan dalam file sistem, dalam database *SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya.

2.2.6 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generated*) oleh komputer[11]. JSON terbuat dari dua struktur yaitu[11]:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

JSON menggunakan bentuk sebagai berikut[11]:

1. Objek

Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup).

Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh “,” (koma).

3. Nilai

Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau *false* atau *null*, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

4. String

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

5. Angka, format oktal dan heksadesimal tidak digunakan.

2.2.7 *Remote Access*

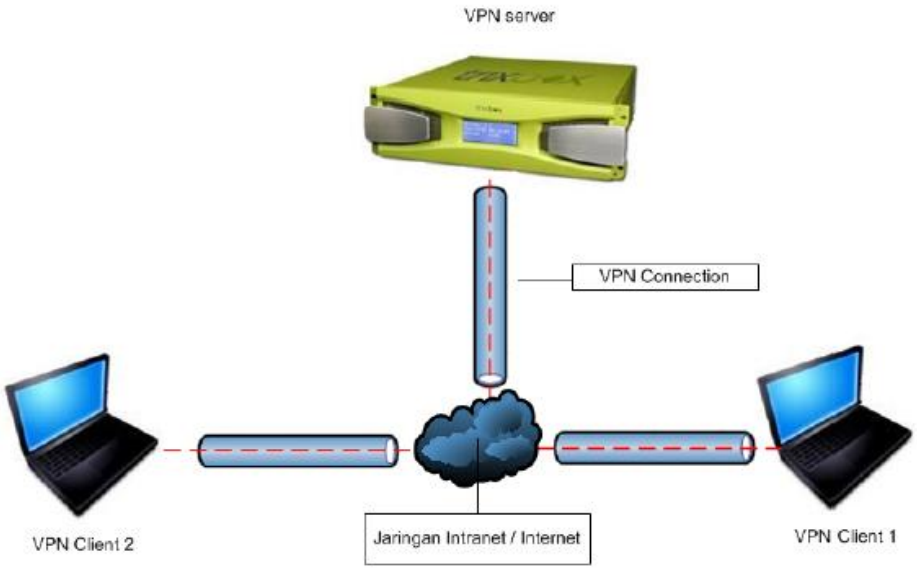
Remote access merupakan mekanisme yang biasa digunakan dalam pengendalian suatu perangkat jaringan, dimana Administrator dapat dengan mudah mengontrol dan mengawasi komputer *server*, *backup data*, atau aktifitas lainnya yang berhubungan dengan *server* [12].

Pada saat ini banyak media yang dapat digunakan untuk dapat melakukan *remote access* ini seperti komputer/laptop, *smartphone* dan perangkat terkomputerisasi lainnya. Untuk dapat melakukan *remote access* ini diharuskan agar perangkat yang akan terlibat *remote* harus terkoneksi dengan internet atau intranet.

2.2.8 *Virtual Private Network* (VPN)

Menurut Andri (2010), VPN (*Virtual Private Network*) adalah teknik pengamanan jaringan dengan cara membuat suatu *tunnel* misalkan pada jaringan

publik atau *internet* sehingga jaringan bersifat *private* dan aman. VPN dikatakan bersifat *private* karena ketika akan terbentuknya sebuah koneksi VPN dibutuhkannya *authentikasi* untuk memastikan bahwa kedua ujung dalam koneksi adalah *user* yang sesuai yang diberi kewenangan untuk mengakses suatu *server*. Setelah terbentuknya suatu koneksi VPN maka data akan di *enkripsi* untuk menjaga kerahasiaan paket yang dikirimkan. Penggunaan *authentikasi* dan *enkripsi* yang dipadukan dalam VPN membuat suatu sistem keamanan yang lebih baik seperti pada Gambar 2.9.



Gambar 2.9 Pelewatan Data dengan VPN

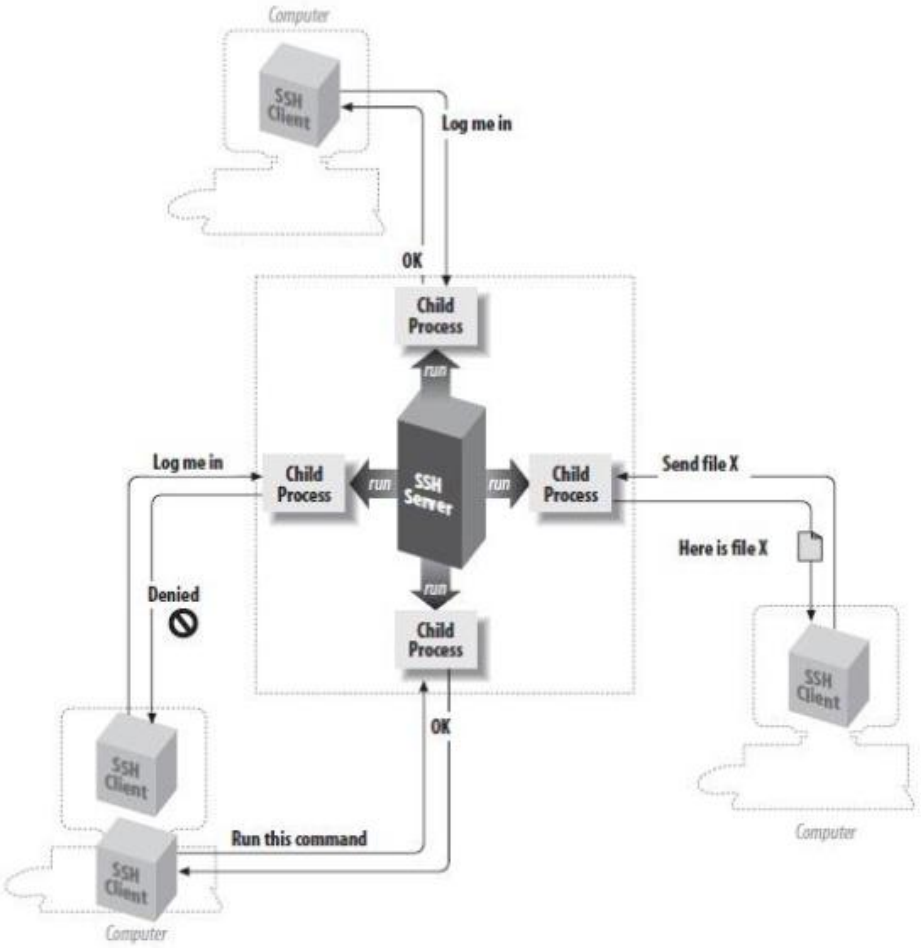
2.2.9 SSH (*Secure Shell*)

SSH (*Secure Shell*) adalah protokol *remote* yang populer berbasis pendekatan yang meminimalisir serangan terhadap suatu jaringan. Setiap kali data dikirimkan kejaringan, SSH secara otomatis meng-enkripsi data atau *command* tersebut. Kemudian ketika data tersebut sampai ke tujuannya, SSH secara otomatis men-dekripsikan data atau *command* tersebut. Hasilnya adalah pengguna dapat bekerja secara normal dan tidak menyadari bahwa komunikasinya telah dienkripsi pada jaringan dengan aman. Metode enkripsi dan algoritma yang

digunakan untuk SSH semua didasarkan pada algoritma standar industri seperti 3DES, *Blowfish, Twofish,* dan AES [13].

SSH memiliki arsitektur *client/server.* Sebuah program *server* SSH biasanya dipasang dan dijalankan oleh Administrator dengan melakukan konfigurasi menerima atau menolak koneksi ke komputer *host.* Administrator kemudian menjalankan SSH-*client* melalui komputer lain untuk mengakses *server* tersebut agar dapat membuat permintaan dari *server* SSH [14].

SSH1 dan protokol SSH1 dikembangkan pada tahun 1995 oleh Tatu Ylonen, peneliti Helsinki University of Technology di Finlandia. Setelah jaringan universitas adalah korban dari serangan *password sniffing* awal tahun itu, Ylonen melecut SSH1 untuk dirinya sendiri. Ketika versi beta mulai mendapatkan perhatian, namun, ia menyadari produk keamanannya dapat dimanfaatkan dengan lebih luas **[**14**]**.



Gambar 2.10 Arsitektur SSH [14]

Protokol SSH bekerja mencakup autentikasi, enkripsi, dan integritas data yang ditransmisikan melalui jaringan [14].

1) Autentikasi

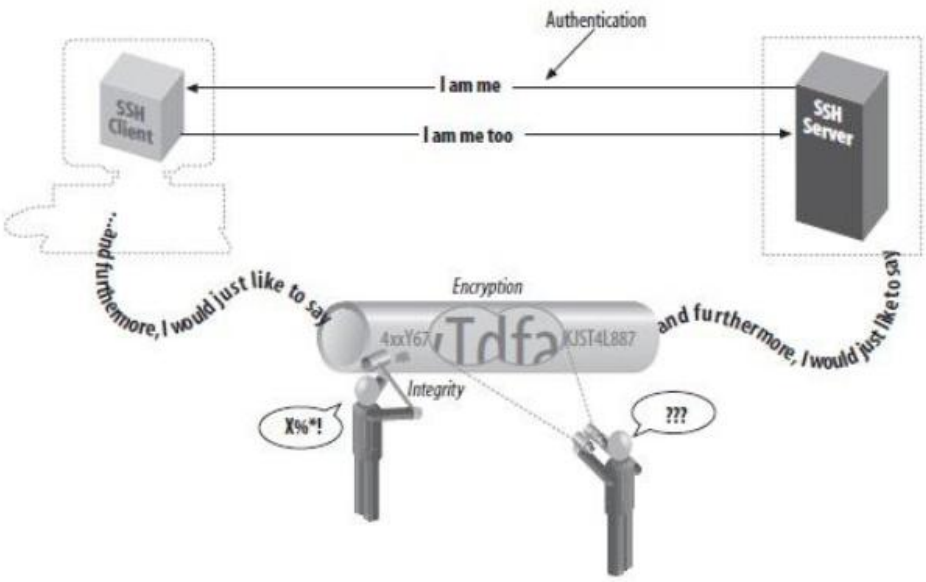
   Jika pengguna mencoba login ke akun komputer *remote*, SSH meminta data identitas pengguna seperti, username dan password. Jika data benar pengguna dapat *login* ke sistem, jika tidak SSH akan menolaknya.

2) Enkripsi

   SSH meng-enkripsi data atau perintah yang dikirimkan dan men-dekripsikannya kepada penerima secara otomatis.

3) Integritas

   Menjamin data atau perintah yang dikirimkan melalui jaringan tidak berubah.



Gambar 2.11 SSH Authentication, Encryption, Integrity [14]

2.2.10 *Unified Modelling Language* (UML)

Pemodelan (*modeling*) adalah proses merancang piranti lunak sebelum melakukan pengkodean (*coding*). Model piranti lunak dapat dianalogikan seperti

pembuatan blueprint pada pembangunan gedung. Membuat model dari sebuah sistem yang kompleks sangatlah penting karena kita tidak dapat memahami sistem semacam itu secara menyeluruh. Semakin komplek sebuah sistem, semakin penting pula penggunaan teknik pemodelan yang baik.

Dengan menggunakan model, diharapkan pengembangan piranti lunak dapat memenuhi semua kebutuhan pengguna dengan lengkap dan tepat, termasuk faktor-faktor seperti *scalability*, *robustness*, *security*, dan sebagainya.

Kesuksesan suatu pemodelan piranti lunak ditentukan oleh tiga unsur, yang kemudian terkenal dengan sebuan segitiga sukses (*the triangle for success*). Ketiga unsur tersebut adalah metode pemodelan (*notation*), proses (*process*) dan tool yang digunakan.

Memahami notasi pemodelan tanpa mengetahui cara pemakaian yang sebenarnya (proses) akan membuat proyek gagal. Dan pemahaman terhadap metode pemodelan dan proses disempurnakan dengan penggunaan tool yang tepat[15].

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*) [15].

2.2.8.1 *Use Case Diagram*

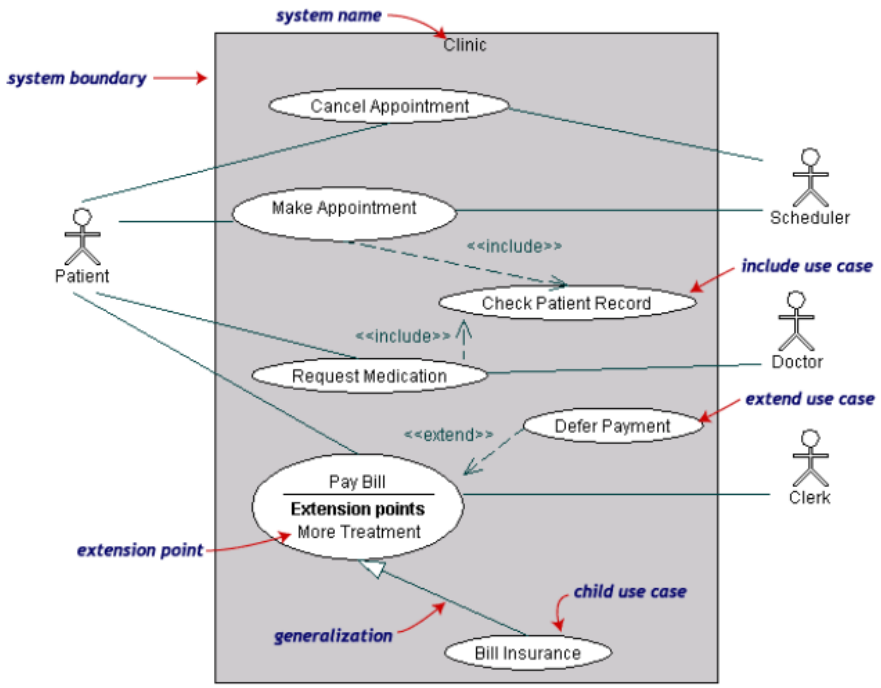
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* 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 melakukan *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 melakukan *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 [15].



Gambar 2.12 Contoh *Use Case Diagram*[15]

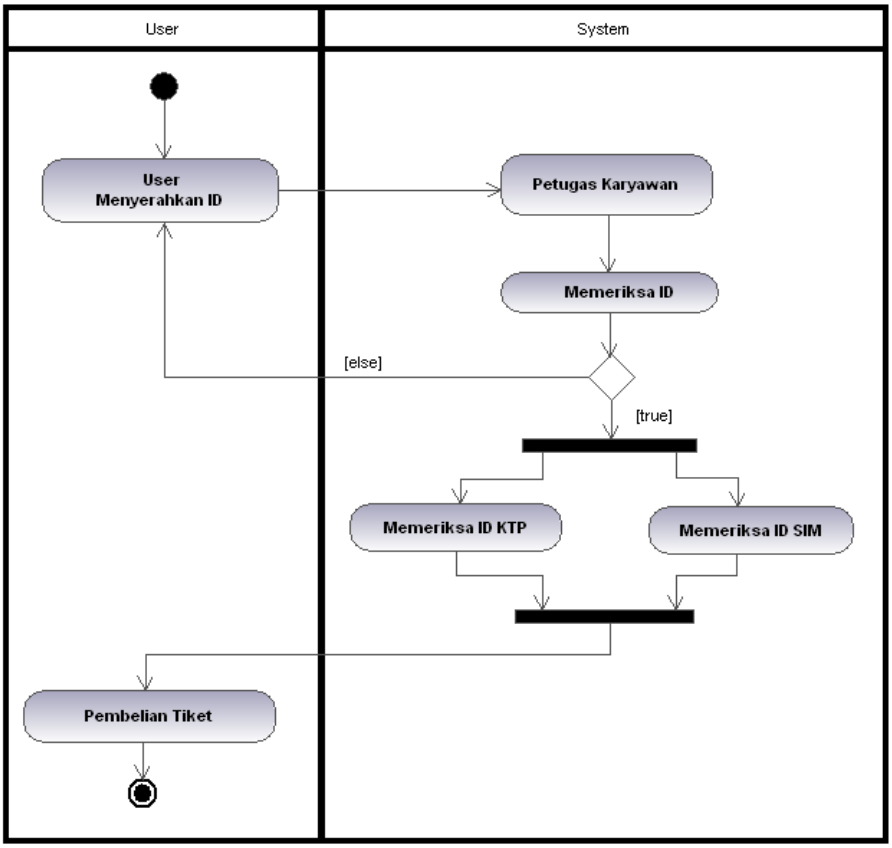
2.2.8.2 *Activity Diagram*

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 *system* (dan interaksi antar sub sistem) 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[15].



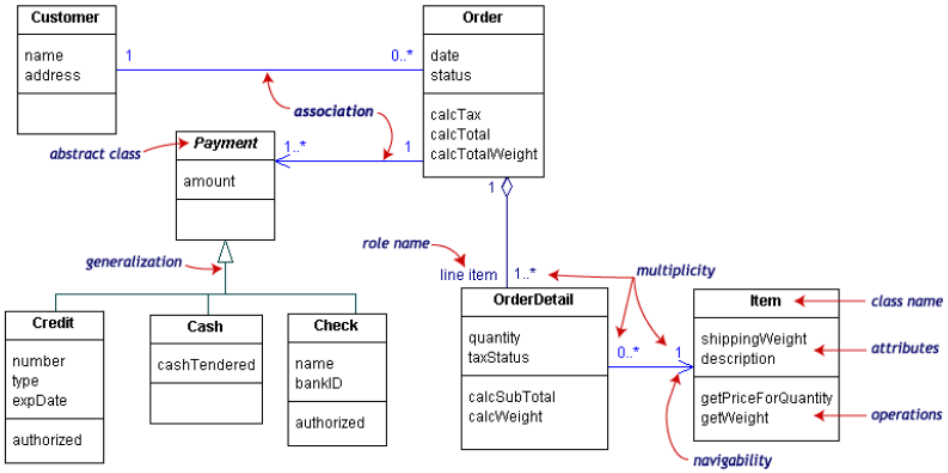
Gambar 2.13 Contoh *Activity Diagram*

2.2.8.3 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.*Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [15].

Class dapat merupakan implementasi dari sebuah interface, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.



Gambar 2.14 Contoh *Class Diagram*[15]

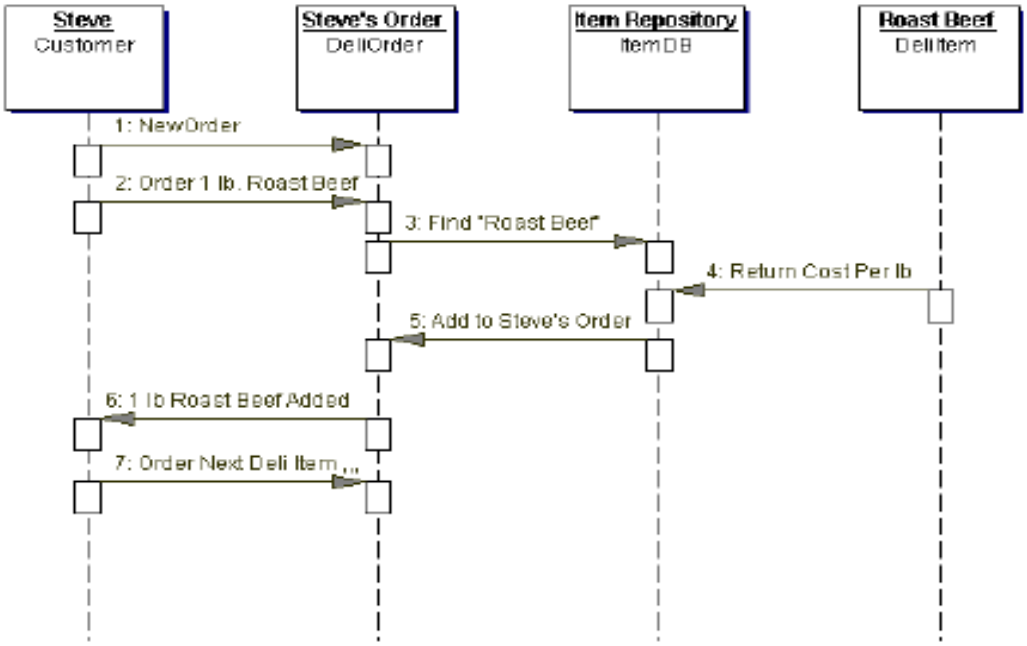
2.2.8.4 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*[15].

Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity* [15].



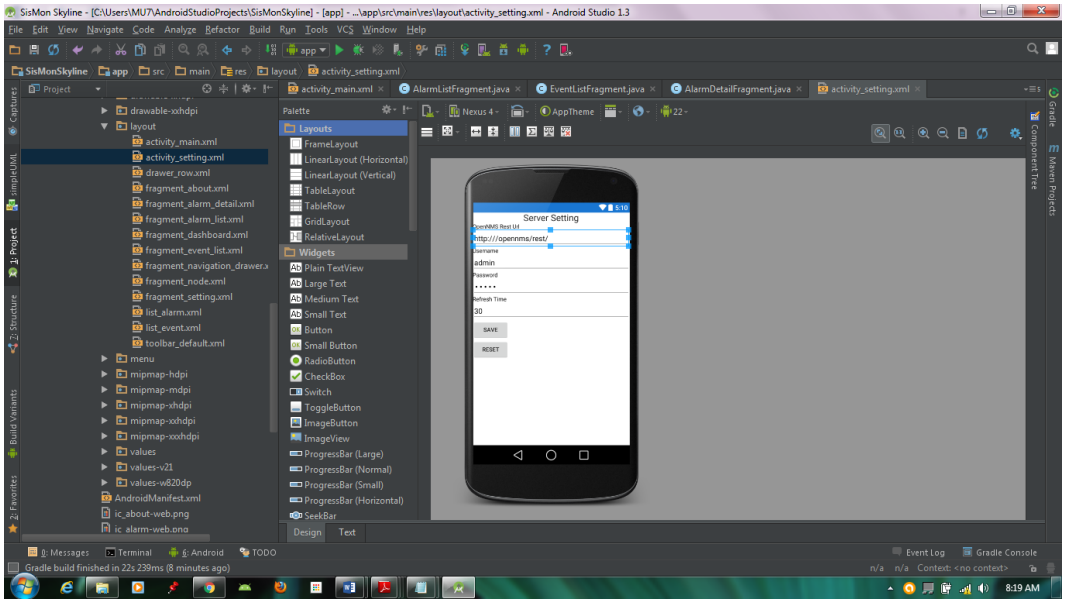
Gambar 2.15 Contoh *Sequence Diagram*

2.2.11 Android Studio

Android Studio adalah sebuah *Integrated Development Environment* (IDE) untuk pengembangan aplikasi di platform Android. Android Studio pertama kali diperkenalkan pada tanggal 16 Mei 2013 oleh Google produk manager di Google I/O Conference. Android Studio menyediakan alat pemrograman Android terintegrasi untuk pengembangan dan *debugging*.

Android Studio didasarkan pada kekuatan, *extensible* IntelliJ IDEA Community Edition, salah satu yang paling canggih dari Java IDE yang tersedia saat ini. Perangkat lunak ini telah ditambahkan dengan fitur yang dirancang dengan mengingat proses pengembangan aplikasi Android. Hal ini akan sangat menyederhanakan dan mengoptimalkan seluruh proses *software* desain dan pengembangan.

Android Studio mempunyai fitur Lint yang merupakan alat untuk mengatasi masalah kompabilitas versi dan *bug* potensial lainnya. Selain itu terdapat fitur ProGuard yang digunakan untuk menghambat *Reverse Engineering*. Perangkat lunak ini juga memiliki Gradle, yang digunakan sebagai alat otomatisasi dalam proses pengembangan perangkat lunak *mobile* Android.



Gambar 2.16 Tampilan Antarmuka Android Studio