BAB II
LANDASAN TEORI

2.1. Konsep Dasar Rekayasa Perangkat Lunak

Perangkat lunak sudah mengalami perkembangan, bahkan sudah mencapai empat dekade dan banyak mendapatkan kelebihan dan kelemahan. Sekarang Perangkat Lunak dikenal sebagai disiplin ilmu yang sah dan layak mendapatkan penelitian dan studi yang sungguh-sungguh serta diskusi yang matang.

2.1.1. Pengertian Perangkat Lunak

Banyak pendapat dari beberapa pihak yang mengartikan pengertian perangkat lunak, diantaranya adalah :

Roger S. Pressman berpendapat bahwa Perangkat lunak adalah:

1. Perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diinginkan.
2. Struktur data yang memungkinkan program memanipulasi informasi secara proporsional.
3. Dokumen yang menggambarkan operasi dan kegunaan program.

2.1.2. Pengertian Rekayasa Perangkat Lunak

Ada beberapa pengertian menurut para ahli di bidang perangkat lunak. Pengertian dari rekayasa perangkat lunak menurut para ahli adalah sebagai berikut:

1. Menurut Stephen R. Schach :

"Rekayasa perangkat lunak adalah sebuah disiplin dimana dalam menghasilkan perangkat lunak bebas dari kesalahan dan dalam pengiriman anggaran tepat waktu serta memuaskan keinginan pemakai."

2. Menurut Fritz Bauer :

   "Rekayasa perangkat lunak adalah penetapan dan penggunaan prinsip rekayasa dalam rangka memperoleh perangkat lunak yang dapat dipercaya dan dapat bekerja secara efisien pada mesin nyata."

3. Menurut IEEE 610.12 :

   "Rekayasa perangkat lunak adalah sebuah studi pendekatan dan aplikasi secara sistematis, disiplin pengembangan operasi dan pemeliharaan perangkat lunak yang kesemuanya itu merupakan aplikasi rekayasa yang berkaitan dengan perangkat lunak."

Dari ketiga pengertian tersebut, arti yang diberikan Fritz Bauer sangat sesuai dengan tujuan dan sasaran dari rekayasa perangkat lunak. Pernyataan Stephen R. Schach terbatas pada menanggulangi kekurangan yang terjadi jika tidak menerapkan rekayasa perangkat lunak. Pengertian yang diberikan IEEE 610.12 paling baik dalam menyampaikan wujud dari rekayasa perangkat lunak.

Rekayasa perangkat lunak merupakan ilmu yang penting untuk diperdalam karena teknologi ini memberikan stabilitas, kontrol, dan organisasi aktifitas yang jika tidak terkontrol dapat menjadi sangat kacau.

Ada beberapa tujuan dilakukan rekayasa perangkat lunak. Tujuan tersebut antara lain :

1. Untuk membangun *software* yang benar dan benar sebuah *software* (*Right Software and Software Right*).
2. Untuk membangun *software* yang tepat (*correct*).
3. Dikelola dengan baik untuk pemeliharaan kebenarannya (*correctness*).

2.2. *Network Monitoring System*

Network Monitoring System menggambarkan sebuah sistem yang terus menerus memonitor jaringan komputer sehingga jika terjadi gangguan dapat secepatnya melakukan notifikasi kepada seorang *network administrator* atau *system administrator*. Sebagai contoh untuk mengetahui status dari sebuah *webserver, software monitoring* secara periodik mengirim *request http;* atau untuk email server, pesan tes di kirimkan melalui sebuah *SMTP* untuk kemudian di ambil melalui *IMAP* ataupun *POP3*.

Yang biasa dijadikan variabel dalam *NMS* ini adalah waktu respon dan ketersediaan *(uptime),* dan konsistensi serta *reliability* juga di perhatikan. Status *request* yang *failure*, seperti ketika koneksi tidak bisa berhubungan *(established),* yang kemudian terputus, yang kemudian sistem monitoring menghasilkan suatu pesan / notifikasi, notifikasi ini bermacam-macam : sebuah *alarm* suara mungkin di kirimkan kepada seorang *net/sys admin*.

2.3. Aplikasi berbasis *web*

Word Wide Web (WWW), lebih dikenal dengan *web*, merupakan salah satu layanan yang didapat oleh pemakai komputer yang terhubung ke internet.

Web pada awalnya adalah ruang informasi dalam internet, dengan menggunakan teknologi *hypertext*, pemakai dituntun untuk menemukan informasi dengan mengikuti *link* yang disediakan dalam dokumen *web* yang ditampilan dalam *browser web.*

Sejalan dengan perkembangan teknologi informasi, aplikasi *web* selain digunakan untuk mendapatkan informasi juga digunakan sebagai sarana untuk mempermudah pekerjaan seorang *administrator* jaringan dalam mengelola jaringan komputer, biasanya aplikasi berbasis *web* digunakan sebagai alat bantu dalam mengkonfigurasi *server* pada jaringan komputer. Berikut adalah beberapa alasan kenapa aplikasi berbasis *web* digunakan:

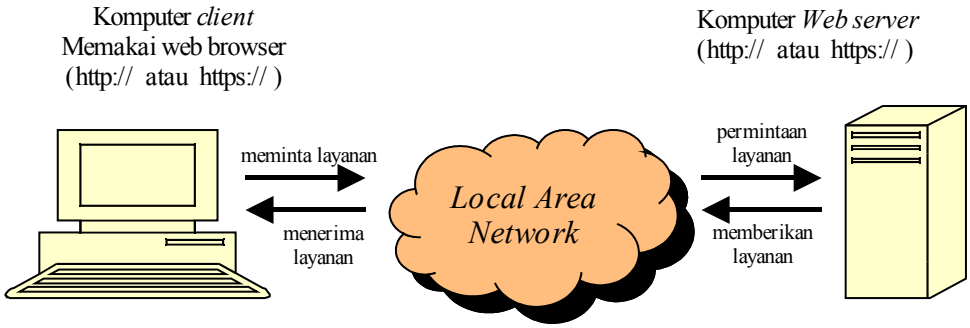
1. Akses informasi mudah.
2. Mempermudah dalam mengkonfigurasi server.
3. Informasi mudah didistribusikan.
4. Bebas platform, dalam hal ini informasi atau pengeksekusian aplikasi dapat dikerjakan atau disajikan oleh *web browser* pada sistem operasi mana saja karena adanya standar dokumen berbagai tipe data dapat disajikan.

Prinsip dasar cara kerja dari aplikasi berbasis *web* adalah sebagai berikut :

1. Aplikasi *web* disimpan dalam dokumen yang disebut dengan halaman-halaman *web (web pages).*
2. *Web page* kemudian disimpan dalam komputer yang disebut dengan *web server,* komputer *web server* ini akan melayani permintaan dari *client* yang akan menggunakan *aplikasi web.*

3. Komputer *client* melakukan eksekusi aplikasi melalui *web browser* yang terdapat pada komputer *client* dengan memasukkan alamat dari aplikasi *web* yang ditempatkan di *web server*.

Pada Gambar 2.1 diperlihatkan cara kerja dari sebuah aplikasi yang berbasis web.



Gambar 2.1 Cara kerja aplikasi berbasis *web*

2.4. *Kannel SMS Gateway*

Kannel SMS Gateway adalah sebuah aplikasi *opensource WAP* dan *SMS gateway* yang sangat handal, yang dipakai di seluruh dunia sebagai *software* penyedia *SMS* dan *WAP*.

Kannel ditemukan oleh Wapit Ltd. Pada bulan Juni, 1999. kemudian wapit tidak lagi meneruskan proyek kannel, dan di lanjutkan dan di koordinasikan oleh anggota dari kannel group diantaranya *the wireless business industry companies Trigenix Ltd (Inggris), Wapme System AG(Jerman), ANAM (IE) dan Global Networks Inc (China).* Yang kemudian The Kannel Group dalam proses pembentukan menuju sebuah badan legal dari proyek kannel ini, yaitu *The Kannel Software Foundation.*

SMS, short message service, banyak digunakan di seluruh dunia dengan pengguna yang sangat banyak. Fungsi utama dari kannel adalah menghubungkan servis berbasiskan *HTTP / web* ke berbagai macam *SMS Center* menggunakan protokol lain.

2.5. *Local Area Network*

Local Area Network (*LAN*) adalah sejumlah komputer yang saling dihubungkan bersama di dalam satu areal tertentu yang tidak begitu luas, seperti di dalam satu kantor atau gedung. Secara garis besar terdapat dua tipe jaringan atau *LAN*, yaitu jaringan *Peer to Peer* dan jaringan *Client-Server*. Pada jaringan *peer to peer*, setiap komputer yang terhubung ke jaringan dapat bertindak baik sebagai *workstation* maupun *server*. Sedangkan pada jaringan *Client-Server*, hanya satu komputer yang bertugas sebagai *server* dan komputer lain berperan sebagai *workstation*. Ciri-ciri jaringan komputer:

1. Berbagi perangkat keras (*hardware*).
2. Berbagi perangkat lunak (*software*).
3. Berbagi saluran komunikasi (*internet*).
4. Berbagi data dengan mudah.
5. Memudahkan komunikasi antar pemakai jaringan.

LAN tersusun dari beberapa elemen dasar yang meliputi komponen *hardware* dan *software*, yaitu :

1. Komponen Fisik

   Personal Computer (*PC*), *Network Interface Card* (*NIC*), *Concentrator* , kabel, konektor, topologi jaringan.

2. Komponen *Software*

   Sistem operasi jaringan, *Network Adapter Driver*, Protokol Jaringan.

2.6. Sistem Operasi *FreeBSD*

FreeBSD adalah *UNIX* like *operating system* (sistem operasi seperti *UNIX*), tersedia gratis di *internet*, sangat banyak digunakan dalam dunia *ISP (Internet service provider), embedded devices,* serta bidang lain yang membutuhkan reliability tinggi.

FreeBSD dibuat dari source code *UNIX* original yang diproduksi oleh AT&T tahun 1970.

Bertahun-tahun yang lalu, AT&T membutuhkan banyak komputer untuk menjalankan bisnis mereka. Saat itu, AT&T dilarang untuk ikut dalam bisnis komputer. Sehingga akhirnya mereka menjual lisensi *software* mereka beserta *source code* nya ke universitas dengan harga yang murah. Mahasiswa universitas yang dapat mengakses teknologi ini dapat membaca *source code* dan belajar bagaimana *software* tersebut bekerja. Sebagai imbalan, AT&T mendapat: programmer gratis, *patch* gratis, serta para *scientist* yang meningkatkan kualitas *software* AT&T. semua orang senang. *Software* AT&T yang sangat terkenal dalam lisensi ini adalah *UNIX*.

Kenapa disebut *UNIX*-like? Ya, kata *"UNIX"* adalah sebuah nama *trademark* (merk dagang) yang dimiliki oleh *The Open Group*. Jadi jika mempunyai produk *operating system*, dan ingin diberi nama *UNIX*, produk tersebut harus disertifikasi oleh *The Open Group*, dan harus membayar uang yang banyak ke mereka. Karena *FreeBSD* dibuat bukan untuk profit, maka namanya menjadi *UNIX*-like.

Jika dibandingkan dengan *OS* yang modern, *UNIX original* tidak terlalu bagus untuk digunakan. Tapi, karena banyak mahasiswa yang mempunyai *source code UNIX*, serta banyak dosen yang mempunyai *project* untuk mahasiswa, *UNIX* mengalami perkembangan cepat. Bertahap, banyak *useful command* dibuat, kemampuan untuk mengontrol *"running program" (job control)* ditambahkan. *Filesystem* ditingkatkan kemampuannya seperti yang kita gunakan sekarang. Perlahan-lahan, seluruh *source code UNIX* original di cabut dan diganti.

Berbagai universitas yang berkerja dengan *UNIX*, berbagi *improvement* yang mereka lakukan dengan *(CSRG UCLA - Computer Systems Research Group, University of California Los Angeles)*. *CSRG* bertindak sebagai sentral *(clearinghouse)* dari *UNIX improvement*.

CSRG mendistribusikan *source code* ini gratis ke semua orang dengan lisensi *AT&T UNIX*.

Hasil pekerjaan *CSRG* adalah koleksi *patches* untuk *UNIX* yang dikenal sebagai *Berkeley Software Distribution*, atau *BSD UNIX*. *Berkeley* juga mendapat donasi *DARPA (Defense Advanced Research Projects Agency)* untuk implementasi *TCP/IP* di *UNIX*.

Proses ini terus berlanjut sampai waktu yang tidak ditentukan. Kalau dilihat copyright statement pada *FreeBSD*, akan telihat seperti ini:

Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
The Regents of the University of California. All rights reserved.

Ya, 15 tahun bekerja – mengembangkan *software* seumur hidup. Kenyataan yang terjadi adalah sebagian besar *source code UNIX original* telah tergantikan &

di *improve* oleh *CSRG* dan para kontributor. Sehingga *source code* asli AT&T terlihat sangat sedikit.

Pada akhirnya, pendanaan *CSRG* berhenti. Setelah beberapa pertikaian politik didalam *University of California.* Pada tahun 1992 *source code* dilepas ke publik dibawah lisensi yang kemudian dikenal sebagai the *BSD license*. Hari ini, *BSD license* mempunyai 3 klausa yang dapat diringkas sebagai berikut:

1. Jangan mengklaim anda menulis *source code.*
2. Jangan salahkan kami jika produk ini rusak/breaks
3. Jangan menggunakan nama kami untuk mempromosikan produk anda.

(pada awalnya, tiap orang yang memakai *software* diharuskan untuk mempunyai *original license*, mereka harus memberitahu *University of California* untuk memakai tersebut. Peraturan ini dihapus beberapa tahun kemudian. Hari ini, orang dapat menggunakan *source code BSD* tanpa harus mengumumkan/memberitahu siapapun).

The BSD license bisa jadi adalah *"the most liberal software license"* yang pernah digunakan. Orang-orang bebas mengambil *BSD* dan memasukkannya kedalam produk *proprietary, free products,* dan *open−source products*. BSD lebih memilih istilah *"copycenter"* daripada *"copyright"* maupun *"copyleft"*. Yang artinya kira-kira seperti ambil *source code* ini dari *copycenter* kemudian bawa *source code* ini kemanapun anda inginkan. Tak heran, perusahaan seperti *Sun Microsystems* juga mengambil dan menjadikannya *copyright* karena memang benar-benar bebas *(free)*.

Selama masa keemasan *CSRG*, pekerjaan *UNIX* dikerjakan dengan cepat di AT&T. AT&T mengambil *source code BSD UNIX distribution* dan

mengintegrasikannya kedalam *UNIX* mereka, kemudian melisensi ulang hasilnya. Proses ini berlangsung mulus sampai pada suatu ketika AT&T tiba-tiba mendapat ijin untuk ikut berkompetisi dalam bisnis software. Mereka mempunyai produk yang sangat tinggi nilai jualnya: sebuah *high-end operating system* yang telah di-*debugged* dan di-develop oleh ribuan orang dari seluruh dunia. Tentu saja mereka sangat senang menjual *UNIX* ini untuk enterprises dengan harga mahal.

BSD license yang diluncurkan tahun1992 membuat kecewa berat *USL (UNIX System Laboratories)* AT&T. Segera, mereka membawa users serta university ke pengadilan. USL mengklaim bahwa *UNIX Berkeley* telah melanggar undang-undang hak kekayaan kekayaan intelektual (HAKI) dengan menyebarkan *source code BSD*. Sementara Berkeley mengklaim bahwa *source code UNIX* tersebut adalah milik mereka, karena mereka telah mekakukan improvement *UNIX* yang sangat signifikan. Pada saat yang sama, telah banyak orang yang telah mengambil *source code BSD* dan menjadikannya produk komersial/*free*. Salah satu dari produk tersebut adalah *386BSD*, yang mana digunakan sebagai inti dari *FreeBSD 1.0*.

Pada tahun 1994, setelah pertengkaran selama dua tahun, kasus ini di tutup setelah pembuktian sebagian besar *source code* AT&T *UNIX* berasal dari BSD. hanya setengah lusin files yang berasal dari AT&T. sebagai solusi dari masalah ini, beberapa file akan didonasikan sedangakan lainnya menjadi *proprietary*. Sayangnya, *FreeBSD 1.X* terdapat beberapa file *proprietary* ini, sehingga banyak pengguna BSD yang bekerja dibayangi ketakutan untuk membangun kembali komponen yang hilang.

Setelah badai berlalu, versi baru *UNIX* di release sebagai *BSD4.4−Lite*. *Update* nya, yaitu *BSD4.4−Lite2*, adalah moyang dari *sourcecode freeBSD* sekarang, juga BSD yang lain seperti *NetBSD, OpenBSD, and Mac OS X.*

Saat ini, *FreeBSD* telah digunakan diseluruh penjuru *internet* Oleh beberapa perusahaan yang berorientasi *internet*. Sebagai contoh, Yahoo! Mempercayakan *FreeBSD* untuk menjalankan hampir seluruh sistem nya. The "baby Bell" US West menggunakan *FreeBSD* untuk menjalankan operasional *internet*. IBM, Nokia, dan banyak perusahaan hardware menggunakan *FreeBSD* pada *embedded systems* (bahkan kamu sendiri tidak tahu ada *FreeBSD* disana). Kenyataannya adalah, jika ada sebuah perusahaan berniat serius untuk melakukan manajemen *bandwidth internet*, kemungkinan besar sistem nya menjalankan *FreeBSD.*

Kekuatan *FreeBSD* :

1. *Portability*

   Tujuan *FreeBSD* adalah menyediakan *OS* yang bebas didistribusikan, dan dapat berjalan pada *hardware* yang populer. Yang saat ini berarti dapat berjalan pada hardware yang dapat dimiliki oleh hampir semua orang *(Intel x86−compatible systems: 386, 486, Pentium I through IV, Celeron, and AMD). FreeBSD* juga berjalan di *processor* Alpha, IA64, AMD64, PowerPC Motorola, juga Sun's SPARC.

2. Kekuatan

   Karena *FreeBSD* dapat berjalan normal pada komputer 386, ia pun juga dapat berjalan baik pada komputer modern. Adalah suatu hal yang menyenangkan jika *OS* yang kita pakai tidak meminta *processor* dan *ram* yang banyak hanya untuk menjalankan *user interface*. Sebagai hasilnya,

anda dapat menggunakan seluruh sumber daya komputer untuk menjalankan tugas-tugas yang anda inginkan. Jika anda memilih menjalankan tugas lewat *GUI, FreeBSD* pun bisa memenuhinya.

3. Management Software yang sederhana

   FreeBSD juga menyederhanakan manajemen *software* melalui koleksi "port"-nya. Umumnya, men-tuning proses pada sistem *UNIX* memerlukan keahlian. "port" menyederhanakan proses ini dengan mengotomatisasi instalasi, *uninstall*, serta proses konfigurasi untuk ribuan paket *software*. (beberapa *BSD OS* lainnya juga membuat membuat sistem paket sendiri yang berbasiskan port).

4. *Optimized Upgrade Process*

   Tidak seperti *Operating system* yang memerlukan prosedur melelahkan dan beresiko, seperti windows. Proses upgrade *FreeBSD* akan membuat sebuah *OS* yang di-optimisasi sesuai aplikasi dan *hardware*. Jika mengganti *hardware*, dapat dengan me-*rebuild* sistem untuk *hardware* tersebut. *vendor* seperti Sun dan Apple juga melakukan seperti ini, karena mereka membuat sendiri *hardware* dan *OS, FreeBSD* tidak dapat memaksa untuk memakai hardware tertentu.

5. *Filesystem*

   Filesystem adalah bagaimana informasi disimpan pada *harddisk* fisik, seperti menyimpan halaman web kedalam bentuk 0 dan 1 di harddisk. *FreeBSD* memiliki *filesystem* yang canggih, dapat mendukung sampai 1 *petabyte* (1.000.000 GB), resisten terhadap kerusakan, serta mempunyai

kecepatan baca-tulis yang tinggi. *BSD filesystem* kemudian diadopsi oleh banyak vendor *UNIX* komersial seperti Sun dan HP.

FreeBSD Development :

1. *Committers*

   Saat ini, *FreeBSD* memiliki hampir 300 *developers*, atau *committers*. *Committers* mempunyai hak *read−and−write* atas *master source code* dan dapat *men-develop, debug*, atau memperbaiki kualitas bagian yang dianggap penting. Jika berminat menjadi developer *FreeBSD,* pastikan anda bergabung dalam mailing list FreeBSD−hackers@FreeBSD.org, yang sebagian berisi diskusi teknis. Beberapa topik, bahkan dipecah lagi dan dibahas dalam milis yang lebih spesifik. Sebagai contoh, development networking dibahas dalam milis FreeBSD−net@FreeBSD.org. ada juga beberapa *channel IRC* tempat kru FreeBSD berdiskusi banyak hal. Para tamu dan pendengar rahasia boleh ikut sepanjang tidak menginterferensi (mengganggu).Chatting internet memang bisa digunakan untuk bermacam-macam tujuan teknis. Para *committers* bertanggung jawab agar *FreeBSD* tetap berjalan, menambah fitur baru, dan mengevaluasi patch yang dikirim oleh *contributor*. Kebanyakan *developers* adalah sukarela, hanya beberapa saja yang dibayar untuk menangani tugas yang rumit dan membuat sakit kepala.

2. *Contributors*

   Sebagai tambahan untuk para committer, FreeBSD mempunyai ribuan *contributor*. *Contributors* tidak perlu khawatir akan merusak *repository* sistem operasi, mereka hanya mengirim patches yang kemudian akan

diterima *comitter*. *Comitter* kemudian meng-evaluasi *patches* tersebut, kemudian memutuskan *patch* mana yang diterima/tidak. Seorang *contributor* yang konsisten dan *patchnya* sering diterima *comitter*, akan ditawarkan untuk menjadi *comitter*.

3. *Users*

   Akhirnya, *FreeBSD* memiliki user yang jauh lebih banyak, susah rasanya untuk menghitung persis berapa jumlah yang *realistik*. Anda dapat mendownload keseluruhan *FreeBSD* dengan *free*, tidak perlu *register*, *upgrade*, atau mengirim *email* ke *mailing list*. Anggap saja kira-kira 5 dari 10 mesin yang ada di *internet* menggunakan *FreeBSD*. Bagian tadi tidak termasuk para komputer-komputer desktop yang berada dibelakang mesin-mesin *FreeBSD*. Jika mesin ini dianggap tidak ada, dan hanya menghitung *server internet*, maka persentasi ini akan naik. Karena *FreeBSD* dapat *menghandle* ratusan bahkan ribuan *domain internet*, maka terjadilah ketimpangan antara situs yang menggunakan *FreeBSD* dibandingan dengan jumlah *server*.

2.7. *Transmission Control Protocol/Internet Protocol (TCP/IP)*

Transmission Control Protocol/Internet Protocol (TCP/IP) adalah sekumpulan protokol yang mengatur komunikasi data komputer di *internet*. Komputer-komputer yang terhubung di *internet* berkomunikasi dengan protokol ini. Karena menggunakan protokol yang sama maka perbedaan jenis komputer dan sistem operasi tidak menjadi masalah.

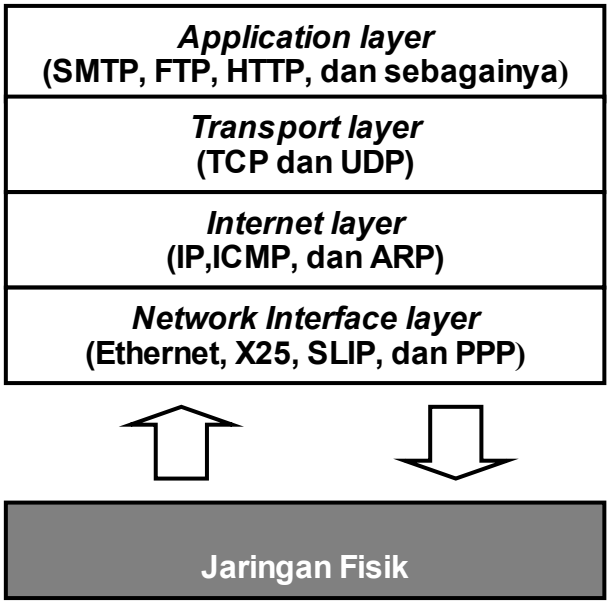
Perkembangan protokol *TCP/IP* yang diterima luas dan praktis menjadi *standar de facto* jaringan komputer berkaitan dengan ciri-ciri yang terdapat pada protokol itu sendiri, yaitu :

1. Protokol *TCP/IP* dikembangkan menggunakan standar protokol terbuka *(open system)*.
2. Standar protokol *TCP/IP* dalam bentuk *Request for Commant (RFC)* dapat digunakan oleh siapapun.
3. *TCP/IP* dikembangkan dengan tidak bergantung pada perangkat lunak atau perangkat keras tertentu.
4. Pengembangan *TCP/IP* dilakukan dengan konsensus dan tidak bergantung pada *vendor* tertentu.
5. *TCP/IP* independen terhadap perangkat keras jaringan dan dapat dijalankan pada jaringan *Ethernet, Token Ring,* jalur telepon *Dial-up* dan praktis jenis media transmisi apapun.
6. Pengalamatan *TCP/IP* bersifat unik dalam skala global. Dengan cara ini, komputer dapat saling terhubung walaupun jaringannya seluas internet sekarang ini.
7. *TCP/IP* memiliki fasilitas *routing* sehingga dapat diterapkan pada *internetwork,* dan memiliki jenis layanan lainnya

2.7.1. *Layer* padaTCP/IP

TCP/IP terbentuk dari empat *layer* (lapisan) kumpulan protokol yang bertingkat yaitu : *Application layer, Transport layer, Internet layer,* dan *Network Interface layer.*

Pada gambar 2.2 diperlihatkan Model *layer* pada protokol *TCP/IP*.



Gambar 2.2. Model *layer* pada *TCP/IP*

Keterangan *layer* pada *TCP/IP*

1. *Network Interface Layer,* bertanggung jawab mengirim dan menerima data ke dan dari media fisik. Media fisiknya dapat berupa kabel *(coaxial/twisted pair), fiber optic,* atau gelombang *electromagnet (*radio/*microwave)*. Karena tugasnya ini, protokol pada lapisan ini harus mampu menerjemahkan sinyal listrik menjadi data digital yang di mengerti komputer, yang berasal dari peralatan yang sejenis.
2. *Internet Layer,* bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat.
3. *Transport Layer,* berisi protokol yang bertanggung jawab untuk mengadakan komunikasi antara dua *host*/komputer. Ada dua buah protokol pada layer ini yaitu *Transmission Control Protocol (TCP)*, menyediakan *service* yang dikenal sebagai *connection oriented*, *reliable*,

byte stream service, dan *User Datagram Protokol (UDP)* bersifat *connectionless*, dalam *UDP* tidak ada *sequencing* (pengurutan kembali) paket yang datang, *acknowledgement* terhadap paket yang datang, atau *retransmisi* jika paket mengalami masalah di tengah jalan.

4. *Application Layer,* pada *layer* ini terletak semua aplikasi yang menggunakan protokol *TCP/IP*

Dalam *TCP/IP*, terjadi penyampaian data dari protokol yang berada di suatu *layer* ke protokol yang berada di *layer* lain. Jika suatu protokol menerima data dari protokol lain di *layer* atasnya, protokol ini akan menambahkan informasi tambahan miliknya ke data tersebut. Setelah itu, data ini di teruskan lagi ke protokol pada *layer* di bawahnya.

2.7.2. Alokasi *IP Address*

IP address adalah alamat yang diberikan pada jaringan komputer dan peralatan jaringan yang menggunakan protokol *TCP/IP*. *IP address* terdiri atas 32 bit angka biner yang dapat dituliskan sebagai empat kelompok angka desimal yang dipisahkan oleh tanda titik seperti 192.168.0.1. *Internet protocol* menggunakan *IP address* sebagai identitas. Pengiriman data akan dibungkus dalam paket dengan label berupa *IP address* si pengirim dan *IP address* penerima. Apabila *IP* penerima melihat pengiriman paket tersebut dengan identitas *IP address* yang sesuai,maka datagram tersebut akan diambil dan disalurkan ke *TCP* melalui port ,dimana aplikasi menunggunya.

IP address terdiri atas dua bagian yaitu *network ID* dan *host ID*, dimana *network ID* menentukan alamat jaringan komputer, sedangkan *host ID* menentukan alamat *host* (komputer, *router*, *switch*). Oleh sebab itu *IP address* memberikan

alamat lengkap suatu *host* beserta alamat jaringan di mana *host* itu berada. Pada gambar 2.3 di perlihatkan bentuk *ip address* dalam bilangan *desimal* dan *biner*.

Network ID			**Host ID**
192	168	0	1
11000000	10101000	00000000	00000001

Gambar 2.3. *IP Address* dalam Bilangan Desimal dan Biner

IP Address dirancang dalam beberapa *class* yang didefinisikan sebagai berikut :

1. Class A :

   Network-Id Host-Id (24 bit)

   0nnnnnnn hhhhhhhh hhhhhhhh hhhhhhhh

2. Class B :

   Network-Id Host-Id (16 bit)

   10nnnnnn nnnnnnnn hhhhhhhh hhhhhhhh

3. Class C :

   Network-Id Host-Id (8 bit)

   110nnnnn nnnnnnnn nnnnnnnn hhhhhhhh

Untuk lebih jelasnya, maka pada tabel 2.1 disajikan *class ip* sebagai berikut.

Tabel.2.1 Perbandingan *class*

Class	**Antara**	**Jumlah Jaringan**	**Jumlah *Host* Per Jaringan**
A	1 s/d 126	126	16.777.214
B	128 s/d 191	16.384	65.534
C	192 s/d 223	2.097.152	254
D	224 s/d 239		
E	240 s/d 255		

Sebuah *Address* khusus didefinisikan dalam *TCP/IP* sebagai alamat *Broadcast*, yaitu alamat yang dapat dikirim kesemua jaringan sebagai upaya *broadcasting*. *Broadcasting IP* diperlukan untuk :

1. Memberikan informasi kepada jaringan, bahwa layanan tertentu *exist*.
2. Mencari informasi dijaringan.

Setiap jaringan *TCP/IP* memerlukan nilai *subnet* yang dikenal sebagai *subnet mask* atau *address mask*. Nilai *subnet mask* memisahkan *network id* dengan *host id*. Pada tabel 2.2 Dapat dilihat perbandingan subnet berikut ini :

Tabel.2.2 Perbandingan *subnet*

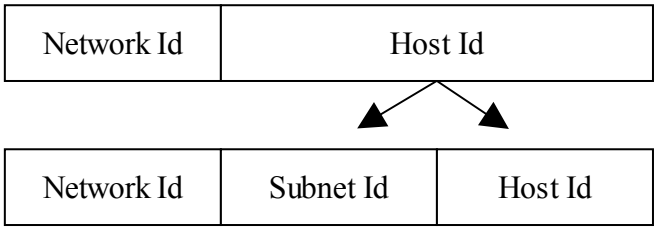
Class	Subnet Mask
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

Subnet mask diperlukan oleh *TCP/IP* untuk menentukan, apakah jaringan yang dimaksud adalah jaringan *local* atau *non local*. Untuk jaringan *non local* berarti harus mentransmisi paket data melalui sebuah *router*. Dengan demikian diperlukan *address mask* untuk menyaring (*filter) IP address* dan paket data yang keluar masuk jaringan tersebut.

2.7.3. *Subnetting IP Address*

Subnet adalah segmen fisik di lingkungan *TCP/IP* yang menggunakan *IP address* yang diturunkan dari *Network-Id* tunggal. Umumnya sebuah organisasi memperoleh satu *Network-Id* dari *interNIC*.

Membagi jaringan menjadi *subnet* memerlukan *setup* segmen menggunakan *Network-Id*, atau *Subnet-Id* yang berbeda. *Subnet-Id* yang unik dibuat untuk setiap segmen dengan membagi bit di *host-Id* menjadi 2 bagian. Satu bagian digunakan untuk mengidentifikasikan segmen sebagai jaringan yang unik, dan bagian lain digunakan untuk mengidentifikasikan *host*. Ini disebut *subnetting* atau *subnetworking*. Pada gambar 2.4 diperlihatkan gambar dari subnet-id.



Gambar 2.4 . *Subnet-Id*

Pada gambar 2.3 berikut adalah tabel yang menunjukkan beberapa altenatif untuk menentukan tingkat *subnetting* yang akan digunakan.

Tabel.2.3 Beberapa alternatif tingkatan *subnetting*

No	Komposisi	Tinjauan 8 bit terakhir	Tinjauan keseluruhan 32 bit
1	hhhhhhhh	seluruh bit tetap menjadi bit host (bit network=0, bit host=8). Kita hanya memiliki 1 buah (2^0) subnetwork yang berkapasitas 256 host (2^8).	Bit network = 24 dan bit host = 8. Tingkat masking adalah 24 bit (netmask 255.255.255.0)
2	nhhhhhhh	bit network=1, bit host=7. Kita memiliki 2 buah (2^1) subnetwork yang berkapasitas 128 host (2^7)	Bit network = 25 dan bit host = 7. Tingkat masking adalah 25 bit (netmask 255.255.255.128)
3	nnhhhhhh	bit network=2, bit host=6. Kita memiliki 4 buah (2^2) subnetwork yang berkapasitas 64 host (2^6)	Bit network = 26 dan bit host = 6. Tingkat masking adalah 26 bit (netmask 255.255.255.192)
4	nnnhhhhh	bit network=3, bit host=5. Kita memiliki 8 buah (2^3) subnetwork yang berkapasitas 32 host (2^5)	Bit network = 27 dan bit host = 5. Tingkat masking adalah 27 bit (netmask 255.255.255.224)
5	nnnnhhhh	bit network=4, bit host=4. Kita memiliki 16 buah (2^4) subnetwork yang berkapasitas 16 host (2^4)	Bit network = 28 dan bit host = 4. Tingkat masking adalah 28 bit (netmask 255.255.255.240)

Keuntungan *Subnetting* :

Dengan menggunakan *subnetting* untuk mengaplikasikan satu *network* pada beberapa segmen fisik, dapat dilakukan :

1. Mencampur beberapa teknologi, seperti *Ethernet* dan *token ring*.
2. Mengatasi keterbatasan teknologi yang ada, seperti melampaui jumlah *host* maksimum per segmen.
3. Mengurangi kongesti jaringan dengan mengarahkan trafik dan mengurangi *broadcast*

Langkah *Subnetting*

1. Tentukan jumlah segmen fisik di jaringan anda
2. Tentukan jumlah alamat host yang diperlukan disetiap segmen fisik. Setiap *host TCP/IP* membutuhkan setidaknya satu *IP address*.
3. Berdasarkan kebutuhan, definisikan : Satu *subnet mask* untuk seluruh jaringan anda, *subnet-Id* yang unik bagi setiap segmen fisik, jangkauan *host-Id* bagi setiap *subnet*.

2.8. *Personal Home Pages (PHP)*

PHP adalah salah satu bahasa *Server-side* yang didesain khusus untuk aplikasi *web*. *PHP* dapat disisipkan diantara bahasa *HTML* dan karena bahasa *Server side,* maka bahasa *PHP* akan dieksekusi di *serve*r, sehingga yang dikirimkan ke *browser* adalah "hasil jadi" dalam bentuk *HTML*, dan kode *PHP* anda tidak akan terlihat.

PHP dahulunya merupakan proyek pribadi dari Rasmus Lerdorf (dengan dikeluarkannya *PHP* versi 1) yang digunakan untuk membuat *home page* pribadinya. Versi pertama ini berupa kumpulan *script perl*. Untuk versi keduanya, Rasmus menulis ulang *script-script PERL* tersebut menggunakan bahasa C, kemudian menambahkan fasilitas untuk *Form HTML* dan koneksi *MySQL*. Setelah mengalami perkembangan oleh suatu kelompok *open source*(termasuk Rasmus), maka mulai versi 3 nya, *PHP* telah menampakkan keunggulannya sebagai salah satu bahasa *server scripting* yang handal. Melalui perkembangan yang pesat ini banyak fasilitas yang ditambahkan dan oleh kelompok ini *PHP* disebut sebagai "*PHP: Hypertext Preprocessor*" . Sintak yang digunakan berasal dari bahasa C , *Java* maupun *Perl*.

PHP merupakan bahasa *script* yang digunakan untuk membuat halaman *web* yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru. Semua *script PHP* dieksekusi pada server dimana *script* tersebut dijalankan. Oleh karena itu, spesifikasi *server* lebih berpengaruh pada eksekusi dari *script PHP* daripada spesifikasi *client*. Namun tetap diperhatikan bahwa halaman *web* yang dihasilkan tentunya harus dapat dibuka oleh *browser*

pada *client*. Dalam hal ini versi dari *html* yang digunakan harus didukung oleh *browser client*.

PHP termasuk dalam *Open Source Product.* Jadi anda dapat merubah *source code* dan mendistribusikannya secara bebas. *PHP* juga diedarkan secara gratis. *PHP* juga dapat berjalan diberbagai *web server* seperti *Apache*, *IIS* dan lain sebagainya.

2.9. *Apache Web Server*

Apache merupakan *web server* yang paling banyak dipergunakan di *Internet*. Program ini pertama kali didesain untuk sistem operasi lingkungan *UNIX*. Namun demikian, pada beberapa versi berikutnya *Apache* mengeluarkan programnya yang dapat dijalankan di *Windows*. Berdasarkan sejarahnya, *Apache* dimulai oleh veteran *developer National Center for Supercomputing Application* (*NCSA) httpd.* Saat itu pengembangan *NCSA httpd* sebagai *web server* mengalami stagnasi. Rob Mc Cool meninggalkan *NCSA* dan memulai sebuah proyek baru bersama para *webmaster* lainnya, menambal *bug*, dan menambahkan fitur pada *NCSA httpd*. Mereka mengembangkan program ini lewat *mailing list*. Dengan berpijak pada *NCSA httpd* versi 1.3, Team *Apache* mengeluarkan rilis pertama kali secara resmi *Apache* versi 0.6.2.

Nama *Apache* diambil dari kata "*A Patchy Web Server*", *server* perbaikan yang penuh dengan tambalan (*patch*). Tambalan yang dimaksud adalah penambahan fitur dan penambalan *bug* dari *NCSA httpd* Versi 1.3. Saat ini *Apache* dipergunakan secara luas. Hal ini disebabkan karena programnya yang gratis, dengan kinerja relatif stabil. Dalam pengembangannya pun mempergunakan sistem

bazaar, yakni tiap orang dibuka kesempatan seluas-luasnya untuk dapat memberikan kontribusi dalam mengembangkan program. Kontribusi dikomunikasikan lewat *mailing list*.

Apache mempunyai program pendukung yang cukup banyak. Hal ini memberikan layanan yang cukup lengkap bagi penggunanya. Beberapa dukungan *Apache* :

1. Kontrol Akses. Kontrol ini dapat dijalankan berdasarkan nama *host* atau nomor *IP*.
2. *Common Gateway Interface (CGI)* , yang paling terkenal untuk digunakan adalah *Practical Extraction and Report Language (Perl)* didukung oleh *Apache* dengan menempatkannya sebagai modul (*mod_perl*).
3. *Personal Home Page (PHP Hypertext Processor)* : program dengan metode semacam *CGI*, yang memproses teks dan bekerja di *server*. *Apache* mendukung *PHP* dengan menempatkannya sebagai salah satu modulnya (*mod_php*). Hal ini membuat kinerja *PHP* menjadi lebih baik
4. *Secure Socket Layer (SSL)* : adalah mekanisme pembungkus data yang berlalu-lalang antara *client* dan *server* dengan suatu enkripsi data. Apache mendukung *SSL* dengan menempatkannya sebagai salah satu modulnya (*mod_ssl*).

2.10. *Secure Socket Layer (SSL)*

SSL adalah protokol pengaman komunikasi antara *client* dan *server*, khususnya pada *e-commerce* (perdagangan elektronik) di internet yang dibuat oleh *Netscape Communications Corp* pada tahun 1994 [2]. *SSL*

melindungi transmisi *HTTP* dengan menambahkan lapisan *enkripsi* pengaman. Protokol *HTTP* yang terkenal untuk *web*, pada dasarnya tidak memiliki pengaman yang handal. *SSL* tidak saja melindungi data yang dikirim melalui *internet* agar tidak dapat dibaca oleh pihak yang berwenang (dengan teknik *enkripsi*), melainkan juga untuk meyakinkan pihak-pihak yang berkomunikasi bahwa lawan bicara mereka di *internet* dapat dipercaya (melalui penggunaan sertifikat digital).

SSL merupakan protokol aplikasi yang independen. Protokol aplikasi yang lebih tinggi (*HTTP*, *FTP*, dan *TELNET*) dapat menggunakan protokol *SSL* secara transparan. *SSL* dapat menegosiasikan algoritma *enkripsi* dan kunci sesi yang akan digunakan antara *client* dan *server*, serta dapat mengotentikasi *server* sebelum terjadi pertukaran data.

SSL dapat memberi keamanan dalam tiga hal :

1. Menjadikan kanal sebagai kanal privat. Enkripsi digunakan terhadap seluruh pesan setelah *handshaking* (protokol pembuka sebelum pertukaran data) yang sederhana digunakan untuk menentukan kunci rahasia. Jadi data-data yang kita kirimkan melalui *internet* ke *server*, misalnya sebuah bank, akan terjamin kerahasiannya. Kriptografi simetri digunakan untuk mengenkrip data (dengan algoritma *DES*, *RC4* dan lain-lain)
2. Kanal diotentikasi. *Server* selalu diotentikasi, sedangkan *client* bisa diotentikasi dan bisa pula tidak. Jadi bila kita membeli buku di Toko online *Amazon.com*, maka kita bisa yakin bahwa yang kita hadapi adalah memang benar-benar *Amazon.com* dan bukannya pihak lain, karena *server* harus diotentikasi (dengan kepemilikan seritifikat digital). Namun di sisi lain,

Amazon.com belum tentu dapat yakin siapa calon pembelinya, karena kita sebagai *client* boleh *tidak* diotentikasi (*client* tidak harus membeli sertifikat digital seperti *server*). Kriptografi asimetri (*RSA, DH*) digunakan untuk otentikasi.

3. Kanal andal, yang berarti bahwa setiap pengubahan data yang sedang dalam perjalanan oleh pihak yang tidak berwenang akan mudah dideteksi dengan penggunaan *Message Integrity (Authentication) Check(MAC*). Fungsi hash yang aman (*MD2,MD5, SHA*) digunakan untuk perhitungan *MAC.*

Protokol *SSL* terdiri dari dua protokol. Yang pertama adalah protokol *SSL Record* yang melapisi protokol transpor (*TCP*) yang andal. Protokol *SSL Record* digunakan untuk membungkus seluruh data yang dikirim dan diterima termasuk protokol *SSL Handshake* (Protokol *SSL* kedua), yang digunakan untuk membangun parameter keamanan sebelum berlangsungnya pertukaran data.

Karena peraturan ekspor yang ketat dari pemerintah Amerika, *SSL* yang diekspor keluar Amerika dibatasi panjang kuncinya hanya 40-bit. Kemudian 128 bit juga diperbolehkan, namun diturunkan dari yang 40 bit tersebut. Tentu saja kekuatannya akan sama dengan yang 40 bit tersebut. Urutan Cara kerja *SSL* diperlihatkan dalam tabel 2.5 berikut ini :

Tabel 2.4 Cara kerja *SSL*

No	Client	Perintah	Server
1	Client Hello	→	
2		←	Server Hello
3		←	Certificate
4		←	Server Hello Done
5	Client Key Exchange	→	
6	Change cipher Spec	→	
7	Fineshed	→	
8		←	Change Cipher Spec
9		←	Finished
10	HTTP request	→	
11		←	HTTP response
12		←	Close Notify Alert
13	Close Notify Alert	→	

Dari tabel 2.5 terlihat bahwa *SSL* bermula dari pengiriman *Hello* oleh *client* ke arah *server*. Jadi jika kita melakukan transaksi e-commerce dengan *Amazon.com*, protokol *SSL* tidak akan bekerja bila kita tidak memulainya (biasanya dengan cara mengklik tombol yang disediakan di halaman *web Amazon.com*). Bila *SSL* sudah beroperasi, maka http:// pada *URL* akan berubah menjadi https:// (*http secure*). Nomor (1-9) merupakan proses handshaking, sedangkan proses pertukaran data yang sebenarnya dimulai dari nomor 10. Jadi setelah persiapan pengamanan (1-9) selesai, maka komunikasi *HTTP* biasa baru dapat beroperasi dengan aman. Karena proses *SSL* yang cukup panjang ini, maka sistem akan menjadi lambat. Karena itu, *SSL* hanya digunakan ketika diperlukan transmisi yang benar-benar aman.

Pada *Client Hello* dan *Server Hello*, *Client* dan *server* menyepakati algoritma enkripsi apa yang akan digunakan beserta perangkat-perangkatnya.

Macam-macam algoritma kriptografi yang digunakan pada *SSL*

SSL CK RC4 128 WITH MD5

SSL CK RC4 128 EXPORT40 WITH MD5

SSL CK RC2 128 CBC WITH MD5

SSL CK RC2 128 CBC EXPORT40 WITH MD5

SSL CK IDEA 128 CBC WITH MD5

SSL CK DES 64 CBC WITH MD5

SSL CK DES 192 EDE3 CBC WITH MD5

Dalam *SSL*, seluruh data yang dikirim akan dibungkus dalam sebuah *record,* yaitu suatu obyek yang terdiri dari *header* dan sejumlah data. Setiap *header record* terdiri dari kode tertentu sepanjang dua atau tiga byte. *Header record* akan dikirimkan sebelum pengiriman data yang sesungguhnya.

Bagian data dari record *SSL* berisi tiga komponen

1. *MAC- DATA[MAC-SIZE]*
2. *ACTUAL-DATA[N]*
3. *PADDING-DATA[PADDING]*

ACTUAL DATA adalah yang diinginkan untuk kita kirimkan. *PADDING-DATA* adalah bit-bit tambahan bila kita menggunakan block cipher dan diperlukan diperlukan tambahan bit. *MAC -DATA* adalah *Message Authentication Code*. *MAC* dihitung sebagai berikut:

MAC-DATA = Hash [SECRET, ACTUAL-DATA, PADDING-DATA, SEQUENCE-NUMBER]

Untuk melakukan otentikasi data, maka data yang sebenarnya (*ACTUAL DATA*) bersama-sama komponen lainnya (*SECRET, PADDING-DATA, SEQUENCE-NUMBER*) dimasukkan ke dalam fungsi *hash* (*MD2,MD5* atau *SHA1*). Pengirim dan penerima sama-sama menghitung nilai ini. Bila berbeda, maka dapat diyakini telah terjadi

pengubahan, khususnya pada *ACTUAL DATA* oleh pihak yang tidak berwenang. Bila penyadap berusaha mengubah *ACTUAL DATA* beserta *MAC-DATA* yang sesuai sekaligus, maka dia tetap akan kesulitan karena penyadap tidak mengetahui nilai *SECRET*.

Bila *client* sedang mengirim pesan ke *server*, maka nilai *SECRET* adalah sama dengan kunci simetri *CLIENT-WRITE-KEY* dan sama juga dengan *SERVER-READ-KEY*. Artinya, kunci *server* yang diberi nama *SERVER*-READ-KEY mempunyai nilai yang sama dengan kunci *client* yang diberi nama *CLIENT-WRITE-KEY*. Jadi apabila *client* sedang mengirim *(WRITE)* maka *server* akan membacanya *(READ)*. Demikian pula sebaliknya bila *server* yang sedang mengirim, maka *SECRET* akan berupa *SERVER-WRITE-KEY* dan akan dibaca oleh *client* dengan kunci *CLIENT-READ-KEY*. Ingat, karena algoritma yang digunakan adalah algoritma kunci simetri, maka baik *client* maupun *server* akan menggunakan kunci yang sama.

2.11. *Perl*

Perl adalah bahasa *scripting* yang diciptakan oleh *Larry Wall* pada tahun 1987. *Perl* menggunakan beberapa fitur dari beberapa bahasa pemrograman yang lain seperti *C, Shell scripting (sh), awk, sed, dan Lisp.*

Secara strukturnya, *perl* berbasiskan gaya bahasa *AWK* dan *C*, dan diadaptasi secara luas karena kekuatannya dalam memproses string, dan karena keterbatasan beberapa s*cript language* yang lain.

Sampai tahun 1991, satu-satunya dokumentasi untuk *perl* adalah sebuah manual page. Pada tahun 1991 buku *"programming perl"* di luncurkan (dikenal oleh

para programmer perl sebagai "camel book") dan menjadi referensi resmi. Pada saat yang bersamaan perl melakukan rilis terbarunya yaitu perl versi 4.

Larry Wall mulai mengerjakan *perl* pada tahun 1987, sementara bekerja sebagai programmer di unisys dan kemudian mengenalkan *perl* versi 1.0 pada *comp.sources.mis* di sebuah *newsgroup* tahun 1987.

Perl versi 5 dirilis pada tahun 1994, dan menjadi sebuah interpreter yang hampir sempurna. Dan menambahkan beberapa fitur termasuk objek, referensi, paket dan modul. Yang paling penting adalah modul yang menyediakan mekanisme memodifikasi bahasa tanpa merubah *interpreter*.

Pada tahun 2007, *perl* masih tetap dimaintain secara aktif, fitur-fitur penting dan konstruksi dasar dari bahasa telah ditambahkan termasuk firtur *unicode* dan *thread*, dan tambahan pada *object oriented programming*. Momen yang paling penting dari perl 5 ini adalah hubungannya dengan modul program. Pada oktober 1995 didirikanlah *"comprehensive perl archive network (CPAN)"*. *CPAN* adalah sebuah arsip *ftp* untuk distribusi modul *perl* dan sumbernya sekaligus. Distribusi *binary* dan dokumentasinya sekaligus.

2.12. *SNMP (Simple Network Management Protocol)*

Simple Network Management Protocol (SNMP) adalah standar manajemen jaringan pada *TCP/IP*. Gagasan di balik *SNMP* adalah bagaimana supaya informasi yang dibutuhkan untuk manajemen jaringan bisa dikirim menggunakan *TCP/IP*. Protokol tersebut memungkinkan *administrator* jaringan untuk menggunakan perangkat jaringan khusus yang berhubungan dengan perangkat jaringan yang lain

untuk mengumpulkan informasi dari mereka, dan mengatur bagaimana mereka beroperasi.

Ada dua jenis perangkat *SNMP*. Pertama adalah *Managed Nodes* yang merupakan node biasa pada jaringan yang telah dilengkapi dengan *software* supaya mereka dapat diatur menggunakan *SNMP*. Mereka biasanya adalah perangkat *TCP/IP* biasa; mereka juga kadang-kadang disebut managed devices. Kedua adalah *Network Management Station (NMS)* yang merupakan perangkat jaringan khusus yang menjalankan *software* tertentu supaya dapat mengatur *managed nodes*. Pada jaringan harus ada satu atau lebih *NMS* karena mereka adalah perangkat yang sebenarnya "menjalankan" *SNMP*.

Managed nodes bisa berupa perangkat jaringan apa saja yang dapat berkomunikasi menggunakan *TCP/IP*, sepanjang diprogram dengan *software SNMP*. *SNMP* didesain supaya host biasa dapat diatur, demikian juga dengan perangkat pintar seperti *router, bridge, hubs,* dan *switch*. Perangkat yang "tidak konvensional" juga bisa diatur sepanjang mereka terhubung ke jaringan TCP/IP: printer, scanner, dan lain-lain.

Masing-masing perangkat dalam manajemen jaringan yang menggunakan *SNMP* menjalankan suatu *software* yang umumnya disebut *SNMP entity*. *SNMP entity* bertanggung jawab untuk mengimplementasikan semua beragam fungsi *SNMP*. Masing-masing *entity* terdiri dari dua komponen utama. Komponen *SNMP entity* pada suatu perangkat bergantung kepada apakah perangkat tersebut managed nodes atau *network management station*.

SNMP entity pada *managed nodes* terdiri atas *SNMP Agent,* yang merupakan program yang mengimplementasikan protokol *SNMP* dan memungkinkan *managed*

nodes memberikan informasi kepada *NMS* dan menerima perintah darinya, dan *SNMP Management Information Base (MIB),* yang menentukan jenis informasi yang disimpan tentang node yang dapat dikumpulkan dan digunakan untuk mengontrol *managed nodes*. Informasi yang dikirim menggunakan *SNMP* merupakan objek dari *MIB*.

Pada jaringan yang lebih besar, *NMS* bisa saja terpisah dan merupakan komputer *TCP/IP* bertenaga besar yang didedikasikan untuk manajemen jaringan. Namun, adalah *software* yang sebenarnya membuat suatu perangkat menjadi *NMS*, sehingga suatu *NMS* bisa bukan *hardware* terpisah. Ia bisa berfungsi sebagai *NMS* dan juga melakukan fungsi lain. *SNMP entity* pada *NMS* terdiri dari *SNMP Manager*: yang merupakan program yang mengimplementasikan *SNMP* sehingga *NMS* dapat mengumpulkan informasi dari *managed nodes* dan mengirim perintah kepada mereka, dan *SNMP Application.* yang merupakan satu atau lebih aplikasi yang memungkinkan *administrator* jaringan untuk menggunakan *SNMP* dalam mengatur jaringan.

Dengan demikian, secara keseluruhan *SNMP* terdiri dari sejumlah *NMS* yang berhubungan dengan perangkat TCP/IP biasa yang disebut *managed nodes*. *SNMP manager* pada *NMS* dan *SNMP agent* pada *managed nodes* mengimplementasikan *SNMP* dan memungkinkan informasi manajemen jaringan dikirim. *SNMP* application berjalan pada *NMS* dan menyediakan interface untuk administrator, dan memungkinkan informasi dikumpulkan dari *MIB* pada masing-masing *SNMP agent*.

2.13. *Snort Intrusion Detection System*

Snort adalah suatu *NIDS (Network-based Intrusion Detection System)*. Sebuah *NIDS* akan memperhatikan seluruh segmen jaringan tempat dia berada, berbeda dengan *host-based IDS* yang hanya memperhatikan sebuah mesin tempat *software host based IDS* tersebut di pasang. Secara sederhana, sebuah *NIDS* akan mendeteksi semua serangan yang dapat melalui jaringan komputer (*Internet* maupun *Intranet*) ke jaringan atau komputer yang kita miliki.

Sebuah *NIDS* biasanya digunakan bersamaan dengan *firewall*, hal ini untuk menjaga supaya *Snort* tidak terancam dari serangan. Sebagai contoh jika *Snort* akan ditempelkan pada *interface ISDN ppp0*, maka sebaiknya di mesin yang sama dipasang *firewall* dan *router* sambungan *dial-up-nya*. Bagi pengguna yang memasang *Snort* pada mesin yang sering sekali diserang, ada baiknya memasang *ACID (Analysis Console for Intrusion Databases),* yang merupakan bagian dari *AIR-CERT project. ACID* menggunakan *PHPlot*, sebuah *library* untuk membuat grafik yang baik di *PHP*, dan *ADODB*, sebuah library abstraksi untuk menggabungkan *PHP* ke berbagai *database* seperti *MySQL* dan *Postgre SQL*. Mengoperasikan *Snort* Secara umum *Snort* dapat dioperasikan dalam tiga mode, yaitu

1. *Sniffer mode*, untuk melihat paket yang lewat di jaringan.
2. *Packet logger mode*, untuk mencatat semua paket yang lewat di jaringan untuk dianalisa di kemudian hari.
3. *Intrusion Detection mode*, pada mode ini *Snort* berfungsi mendeteksi serangan yang dilakukan melalui jaringan komputer. Untuk menggunakan mode IDS ini diperlu kan setup dari berbagai aturan yang akan membedakan sebuah paket normal dengan paket yang membawa serangan.

Snort selain terdapat pada Linux, kini terdapat pula pada Windows lengkap dengan *GUI (Graphical User Interface) IDS Center* yang amat memudahkan penggunaannya. Snort memanfaatkan *WinPCap* untuk menangkap paket paket data yang lalu-lalang melalui jaringan.

2.14. *Cron*

cron merupakan program *daemon* yang setiap menit men-cek apakah ada perintah yang mesti dijalankan. Daftar atau tabel *cron (crontab)*, yaitu file yang berisi jadwal perintah, ada di direktori *spool (misalnya /var/spool/cron/)*. Nama file itu sama dengan nama user pemilik atau pembuat jadwal. Untuk membuat atau mengedit crontab, user dapat menggunakan program yang juga bernama crontab. Secara *default, crontab* akan menjalankan editor yang dijadikan standar, biasanya vi. Sebagai root, kita dapat mengedit langsung file crontab yang ada di direktori spool cron.

Ada 3 *option* dalam menjalankan *crontab* sebagai *user* biasa:

1) $ crontab -e (artinya mengedit atau membuat baru tabel cron atau daftar tugas).
2) $ crontab -l (artinya menampilkan daftar tugas yang telah ada).
3) $ crontab -r (artinya menghapus daftar tugas yang ada).

Sebagai root, kita dapat mengedit, melihat atau menghapus crontab user dengan tambahan option -u, misalnya:

crontab -eu rus (artinya mengedit crontab yang dimilki user rus).

Sebuah crontab merupakan daftar program yang akan dijalankan secara berulang melalui terminal atau command line, lengkap dengan waktu kapan program

tersebut dijalankan. Setiap program dijadwalkan dalam satu baris dengan makna sebagai berikut: menit jam tanggal bulan hari

Contoh crontab di bawah ini akan menjalankan fetchmail setiap menit (*), setiap jam (*), setiap hari/tanggal (*), setiap bulan (*) dan setiap hari dalam seminggu (*).

* * * * * fetchmail

Contoh berikut ini, fetchmail hanya dijalankan setiap (interval) 5 menit (*/5), antara jam 8 pagi hingga jam 5 sore (8-17), setiap hari Senin hingga Jumat (1-5).

*/5 8-17 * * 1-5 fetchmail

Nama *(case sensitive)* juga dapat digunakan untuk crontab, misalnya sun-sat (artinya Minggu hingga Jumat), jan-dec (Januari – Desember).

Izin Menggunakan crontab Ada 2 file yang berfungsi mengatur pemberian izin penggunaan crontab, yaitu /etc/cron.allow dan /etc/cron.deny. Yang dicek pertama adalah cron.allow, yaitu daftar user yang diizinkan menggunakan crontab. User selain yang terdaftar di cron.allow tidak boleh menggunakan crontab. Bila cron.allow tidak ada, cron.deny akan dibaca. User selain yang terdaftar di cron.deny diizinkan menggunakan crontab. Jika kedua file tidak ada, semua user dapat menggunakan crontab.