BAB II

LANDASAN TEORI

2.1 TCP/IP

Dalam arti yang sederhana, TCP/IP *(Transmission Control Protocol/Internet Protocol)* adalah nama keluarga protokol jaringan. Protokol adalah sekelompok aturan yang harus diikuti oleh perusahaan-perusahaan dan produk-produk software agar produk mereka itu bisa kompatibel satu dengan yang lainnya. Suatu protokol menentukan bagaimana suatu software berkomunikasi dengan software lainnya, juga menentukan bagaimana setiap bagian dari keseluruhan paket menentukan bagaimana perjalanan informasinya.

Internet, memang dari awal sudah dibuat dengan menggunakan TCP/IP pada tingkat protokolnya yang memungkinkan sistem apapun yang terhubung kedalamnya bisa berkomunikasi dengan sistem lain tanpa mempedulikan bagaimana sistem masing-masing bekerja.

TCP/IP sebenarnya adalah dua macam protokol berbeda. Tidak seperti di anggap kebanyakan orang, istilah TCP/IP mengacu ke seluruh keluarga protokol yang dirancang untuk mentransfer informasi sepanjang jaringan. TCP/IP dirancang untukl menjadi komponen perangkat lunak dari suatu jaringan. Semua bagian didalam keluarga TCP/IP memiliki tugas tersendiri, misalnya mengirim e-mail, mentransfer file, menyediakan layanan *login* jarak jauh *(remote login)* dan menangani informasi *routing* jaringan.

Protokol TCP bertanggung jawab memecah informasi kedalam beberapa paket, sedang IP bertanggung jawab mengangkut (mentransfer) paket-paket tersebut sesuai tujuan. Kemudian TCP bertugas menyatukan kembali paket-paket tersebut ke urutan yang benar.

Layanan dalam TCP/IP yang berbeda dikelompokkan menurut fungsi-fungsinya. Protocol-protokol transport mengendalikan pergerakan data antara dua mesin, mencakup :

1. TCP *(Transmission Control Protocol)*

   Protokol ini bersifat *connection-based*, artinya kedua mesin pengirim dan penerima tersambung dan berkomunikasi antara satu sama lain sepanjang waktu.

2. UDP *(User Datagram Protocol)*

   Protokol ini bersifat *connectionless* (tanpa koneksi), artinya data dikirim tanpa mesin penerima dan pengirim saling berhubungan. Ini seperti mengirim surat lewat kantor pos, surat dikirim oleh si pengirim namun ia tidak pernah tahu apakah surat itu sampai di tujuan atau tidak.

Sementara itu adapula protokol-protokol *routing* untuk menangani pengalamatan *(addressing)* data dan menentukan jalur terbaik untuk mencapai tujuan. Protokol-protokol tersebut juga bertanggung jawab memecah informasi-informasi ukuran besar dan menyusunnya kembali pada tujuan. Protokol-protokol tersebut antara lain :

1. IP *(Internet Protocol)* menangani transmisi data yang sebenarnya.

2. ICMP *(Internet Control Message Control Protocol)* menangani informasi status untuk IP, seperti *error* (kesalahan) dan perubahan-perubahan dalam perangkat keras jaringan yang mempengaruhi *routing* (penentuan jalur).
3. RIP *(Routing Information Protocol)* dan OSPF *(Open Shortest-Path First),* yaitu satu dari berbagai protokol yaqng menentukan metode *routing* terbaik untuk menyampaikan data.

2.2 World Wide Web

Pada awalnya Internet adalah sebuah proyek yang dimaksudkan untuk menghubungkan para ilmuan dan peneliti di Amerika, namun saat ini telah tumbuh menjadi media komunikasi global yang dipakai semua orang di muka bumi. Pertumbuhan ini membawa beberapa masalah penting mendasar, diantaranya kenyataan bahwa Internet tidak diciptakan pada jaman *graphical user interface* (GUI) seperti saat ini. Internet dimulai pada masa dimana orang masih menggunakan alat-alat akses yang tidak *user-friendly* yaitu terminal berbasis teks serta perintah-perintah *command line* yang panjang-panjang serta sukar diingat, sangat berbeda dengan komputer dewasa ini yang menggunakan klik tombol mouse pada layer grafik berwarna.

Kemudian orang mulai berfikir untuk membuat sesuatu yang lebih baik. Popularitas Internet mulai berkembang pesat seperti jamur di musim penghujan setelah standar baru yaitu HTTP *(Hypertext Transfer Protocol)* membuat pengaksesan informasi melalui protocol TCP/IP menjadi lebih mudah dari

sebelumnya. HTML *(Hypertext Markup Language)* memungkinkan orang menyajikan informasi secara visual lebih menarik. Pemunculan HTTP dan HTML kemudian membuat orang mengenal istilah baru dalam Internet yang sekarang menjadi sangat popular, bahkan sedemikian populernya sehingga identik dengan Internet itu sendiri, yaitu *Worl Wide Web* (WWW).

Pada prinsipnya *Worl Wide Web* (singkatnya cukup disebut "web" saja) bekerja dengan cara menampilkan file-file HTML yang berasal dari server web pada program *client* khusus, yaitu *browser web*. Program *browser* pada *client* mengirimkan permintaan *(request)* kepada *server web*, yang kemudian akan dikirimkan oleh *server* dalam bentuk HTML. File HTML berisi instruksi-instruksi yang diperlukan untuk membentuk tampilan. Perintah-perintah HTML ini kemudian diterjemahkan oleh *browser web* sehingga isi informasinya dapat ditampilkan secra visual kepada pengguna di layar komputer.

Gambar 2.1 Konsep dasar browser dan server web

2.3 Hypertext Transfer Protocol

Web merupakan terobosan baru sebagi teknologi sistem informasi yang menghubungkan data dari banyak sumber dan layanan yang beragam macamnya di Internet. Pengguna tinggal mengklikkan mousenya pada link-link hypertext yang ada untuk melompat ke dokumen-dokumen di berbagai lokasi di Internet. Link-linknya sendiri bisa mengacu kepada dokumen web, server FTP (*File Transfer Protocol)*, e-mail ataupun layanan-layanan lain.

Server dan browser web berkomunikasi satu sama lain dengan protokol yang dibuat khusus untuk ini, yaitu HTTP. HTTP bertugas menangani permintaan-permintaan *(request)* dari browser untuk mengambil dokumen-dokumen web.

HTTP bisa dianggap sebagai sistem yang bermodel *client-server*. *Browser* web sebagai clientnya, mengirimkan permintaan kepada *server* web untuk mengirimkan dokumen-dokumen web yang dikehendaki pengguna. *Server* web lalu memenuhi permintaan ini dan mengirimkannya melalui jaringan kepada *browser*. Setiap permintaan akan dilayani dan ditangani sebagi suatu koneksi terpisah yang berbeda.

Semua dokumen web dikirim sebagai file teks biasa. Sewaktu mengirimkan request kepada *server* web, *browser* juga mengirimkan sedikit informasi tentang dirinya, termasuk jenis-jenis file yang bisa dibaca olehnya. Informasi ini lalu digunakan oleh *server* web untuk menentukan apakah dokumen yang diminta bisa dikirimkan kepada *browser* atau tidak.

HTTP bekerja di atas TCP *(Transmission Control Protocol)* yang menjamin sampainya data ditujuan dalam urutan yang benar. Bila suatu kesalahan terjadi selama proses pengiriman, pihak pengirim akan mendapatkan pemberitahuan bahwa telah terjadi ketidakberesan. Karenanya *server* dan *client* tidak harus menyediakan mekanisme untuk memeriksa kesalahan transmisi data, yang berarti mepermudah pengerjaan pemrograman. Namaun demikian, HTTP tidak memiliki apa yang disebut *session*, seperti halnya FTP, yang menjaga hubungan antara *server* dan *client* secarea konsisten. Setiap halaman web yang dikirim akan melibatkan satu proses penyambungan antara *client* dan *server*, baru kemudian datanya ditransfer, koneksi antara *server* dan *client* diputus. Sifat ini membuat HTTP sering disebut dengan istilah *hit-and-run.*

Suatu halaman web seringkali berisi beberapa file gambar, atau beberapa file-file lain. HTTP memaksa *server* untuk menjalin hubungan baru setiap kali hendak mengirim satu buah file. Ini tidak efisien dan menguntungkan, mengingat proses hubung-putus-hubung semacam ini menyebabkan beban bagi jaringan.

Standar baru protokol HTTP, yaitu HTTP/1.1 dirancang untuk mengatasi masalah diatas. Web diarahkan agar mengarah ke penggunaan *persistent connection* (sambungan yang terjaga berkesinambungan) secara lebih efisien. Dalam HTTP/1.1, *server* tidak akan memutuskan hubungan dengan *client* pada akhir pentransferan dokumen. Hubungan tetap dibuka untuk melayani bila saja ada *request* lagi dalam waktu yang singkat. Hubungan baru akan diputus jika melewati suatu batas waktu tertentu (yang bisa ditentukan oleh *administrator server*) *client* tidak mengirimkan *request* lagi.

Keuntungan lain dari *persistent connection* adalah penggunaan *pipelining*. *Pipelining* adalah proses pengiriman request berikutnya segera setelah *reques*t sebelumnya dikirimkan tanpa menunggu balasan dari *server* terlebih dahulu. *Server*nya tetap harus melayani setiap *request* secara berurutan, namun ini mengurangi waktu tunda antara setiap *request*. Hasilnya data akan lebih cepat sampai tujuan.

2.4 Hypertext Markup Language

HTML dewasa ini dikenal sebagai bahasa standar untuk membuat dokumen web. Sesungguhnya *Hypertext Markup Language* (HTML) justru tidak dibuat untuk mepublikasikan informasi di web, namun oleh karena kesederhanaan serta kemudahan penggunaannya, HTML kemudian dipilih orang untuk mendistribusikan informasi di web.

Perintah-perintah HTML diletakkan dalam file berekstensi *.html dan ditandai dengan mempergunakan *tag* (tanda) berupa karakter “<” dan “>”. Tidak seperti bahasa pemrograman berstruktur procedural seperti Pascal atau C, HTML tidak mengenal *jumping* ataupun *looping*. Kode-kode HTML dibaca oleh *browser* dari atas ke bawah tanpa adanya lompatan-lompatan.

Struktur sebuah dokumen HTML pada dasarnya dibagi menjadi dua bagian besar, yaitu *header* dan *body*. Masing-masing ditandai oleh pasangan container *tag* <head> dan <body>. Bagian *head* berisikan judul dokumen dan informasi-informasi dasar lainnya. Sedangkan bagian *body* adalah data

dokumennya. Pengaturan format teks dan pembentukan link dilakukan terhadap obyeknya langsung dengan ditandai oleh tag-tag HTML.

HTML diatur oleh konsorsium WWW (W3C). semua perubahan atas standar bahasa HTML harus disahkan terlebih dahulu oleh konsorsium ini. Sejauh ini, HTML telah mengalami bebagai revisi sepanjang hidupnya. Standar terakhir yang sekarang diperkenalkan adalah standar HTML 4.0, yang mendukung antara lain CSS *(cascading style sheet), dynamic content positioning* (penempatan isi secara dinamis), *downloadable font*(jenis font yang bisa didownload secara otomatis) dan sebagainya. Hingga kini, tidak semua *browser* web telah disesuaikan untuk mendukung standar HTML terbaru ini, sehingga banyak masalah inkompabilitas antara macam-macam *browser* web.

2.5 Browser dan Server Web

Dalam dunia web, perangkat lunak *client*, yaitu *browser* web mempunyai tugas yang sama yaitu menterjemahkan informasi yang diterima dari *server* web dan menampilkan pada layar komputer pengguna. Oleh karena HTTP memungkinkan *server* web mengirimkan beragam data, seperti teks atau gambar, *browser* harus bisa mengenali berbagai macam data yang akan diterimanya, dan selanjutnya harus tahu cara untuk menampilkannya dengan benar. Teks harus ditampilkan sebagai teks dan gambar harus ditampilkan sebagai gambar.

Umumnya *browser* web menerima data dalam bentuk HTML. File HTML sebenarnya adalah file teks biasa yang selain berisi informasi yang hendak ditampilkan kepada pengguna, juga mempunyai perintah-perintah untuk mengatur

tampilan data tersebut. *Browser*lah yang memiliki kuasa penuh dalam menterjemahkan perintah-perintah tadi. Meskipun sudah dibuat kosensus untuk menstandarkan format dan elemen-elemen HTML, setiap jenis *browser* bisa menterjemahkan file HTML yang sama secara berbeda.

Pertamakali protokol-protokol dasar web dikembangkan yaitu sekitar tahun 1990-an, *browser* pertama yang dikenalkan adalah mosaic yang dibuat oleh *National Center for Supercomputing Aplications* (NCSA) di Amerika Serikat. Mosaic dimaksudkan agar menjadi sebuah interface grafis yang mudah dipergunakan, yang demikian diharapkan dapat mempercepat perkembangan dan dukungan umum akan web. Mosaic langsung dibuat untuk tiga flatform berbeda, yaitu X Window (untuk lingkungan UNIX dan keluarganya), Microsoft Windows dan Macintosh. Mosaic inilah yang lalu dianggap sebagai legenda yang memacu revolusi web menjadi sedemikian populernya seperti sekarang ini.

Perkembangan jaman serta semakin populernya lingkungan GUI *(Graphical User Interface)* membuat banyak orang sekarang berlomba-lomba untuk membuat program *browser* yang menarik serta mudah dipakai. *Browser-browser* web modern dilengkapi dengan fasilitas-fasilitas yang mendukung tampilan multimedia berupa audio (suara), animasi tiga dimensi, bahkan video. Program *browser* yang paling terkenal saat ini adalah Nescape Navigator dan Microsoft Internet Explorer.

Sementara itu *server* web pada dasarnya adalah perangkat lunak khusus yang bertugas melayani permintaan-permintaan dari *browser* web akan dokumen-dokumen yang tersimpan di dalamnya. Perangkat lunak *server* web sekarang telah

tersedia untuk berbagai macam platform dan lingkungan sistem operasi untuk lingkungan UNIX, yang paling popular adalah Apache, Nescape FasTrack dan NCSA HTTPD. Sementara untuk lingkungan Windows tersedia Microsoft Internet Information Server (IIS), Nescape FasTrack, O'Reilly Website dan banyak lagi. Sistem operasi jaringan Novell Netware pun memiliki suatu modul *add-on* yang berfungsi sebagai *server* web, yang bisa dijalankan pada saat strtup jaringan.

2.6 Apache Web Server

Apache merupakan salah satu contoh *web server* yang paling banyak digunakan saat ini. Tugas dari sebuah web server adalah menyajikan informasi dalam bentuk halaman-halaman web beserta kelengkapannya.

Beberapa perangkat lunak *server* web mempunyai feature seperti *server side programming, security control* dan lain sebagainya. Meskipun beragam macamnya, secara fungsional semua jenis *server* web adalah sama saja, yaitu berfungsi melayani permintaan-permintaan dari *browser* web.

2.7 PHP

2.7.1 Tentang PHP

PHP adalah teknologi yang diperkenalkan tahun 1994 oleh Rasmus Lerdorf. Beberapa versi awal yang tidak dipublikasikan digunakan pada situs pribasdinya untuk mencatat siapa saja yang mengakses daftar riwayat online-nya. Versi pertama digunakan oleh pihak lain pada awal tahun 1995 dan dikenal

sebagai Personal Home Page Tools. Terkandung didalamnya sebuah *parser engine* (mesin pengurai) yang sangat disederhanakan, yang hanya mampu mengoleh makro khusus dan beberapa utilitas yang sering digunakan pada pembuatan Home Page, seperti buku tamu, pencacah, dan hal semacamnya. Parser tersebut ditulis ulang pada pertengahan 1995 dan dinamakan PHP/FI Version 2.FI (Form Interpreter) sendiri berasal dari kode lain yang ditulis juga oleh Rasmus, yang menterjemahkan HTML dari data. Ia menggabungkan script Personal Home Page Tools dengan form interpreter dan menambahkan dukungan terhadap *server* database yang menggunakan format MSQL, sehingga lahirlah PHP/FI. PHP/FI tumbuh dengan pesat, dan orang-orang mulai meyiapkan kode-kode programnya supaya bisa didukung oleh PHP.

Sulit memberikan data statistic yang akurat, namun diperkirakan pada akhir 1996 PHP/FI sudah digunakan sedikitnya pada 15.000 situs web diseluruh dunia. Pada pertengahan 1997, angka tersebut berubah menjadi 50.000. pada saat itu juga terdapat perubahan dalam pengembangan PHP. PHP berubah dari proyek pribadi Rasmus berubah menjadi sebuah tim yang lebih terorganisasi. Parsernya ditulis ulang dari bentuk rancangan awal oleh Zeev Suraski dan Andi Gutmans, dan parser baru ini adalah sebagai dasar PHP Version 3. Banyak kode utilitas yang berasal dari PHP/FI diport ke PHP3, dan banyak diantaranya sudah selesai ditulis ulang secara lengkap.

Pada pertengahan 1998, baik PHP/FI maupun PHP3 dikemas bersama dengan produk-produk komersial seperti *server* web StrongHold buatan C2 dan Linux RedHat, dan membuat survei yang dilakukan oleh NetCraft, kemungkinan

PHP digunakan pada lebih dari 150.000 situs web diseluruh dunia. Sebagai pembanding, angka tersebut lebih banyak daripada pengguna server web Enterprise Server buatan Netscape di Internet.

PHP (*PHP Hypertext Preprocessor*), adalah sebuah bahasa scripting yang dibundel dengan HTML, yang dijalankan disisi server. Sebagian besar perintahnya berasal dari C, Java dan Perl dengan beberapa tambahan fungsi khusus PHP. Bahasa ini memungkinkan para pembuat aplikasi web menyajikan halaman HTML dinamis dan interaktif dengan cepat dan mudah, yang dihasilkan server. PHP juga dimaksudkan untuk mengganti teknologi lama seperti CGI (Common Gateway Interface).

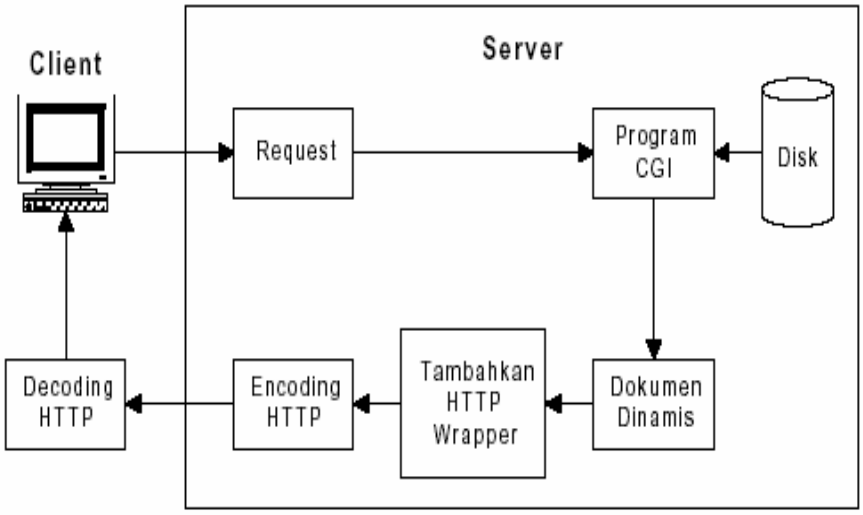
PHP bisa berinteraksi dengan hamper semua teknologi web yang sudah ada. Developer bisa menulis sebuah program PHP yang mengeksekusi suatu program CGI di server web lain. Fleksibilitas ini amat bermanfaat bagi pemilik situs-situs web yang besar dan sibuk, karena pemilik masih bisa mempergunakan aplikasi-aplikasi yang sudah terlanjur dibuat di masa lalu dengan CGI, ISAP, atau dengan Sript seperti Perl, Awk atau Python selama proses migrasi ke palikasi baru yang dibuat dengan PHP. Ini mempermudah dan memperhalus peralihan antara teknologi lama dan teknologi baru.

2.7.2 Web Dinamis

Saat server melayani permintaan dari browser web akan suatu dokumen, server sebenarnya hanya menmgambil suatu file di dalam disk dan melakukan beberapabeberapa pekerjaan untuk transisi seperti menambahkan informasi tipe

dokumen, merubah formatnya agar bisa dikirim menggunakan HTTP, yang mengirimkan semuanya ke browser. Browser web menerima file HTML dan menampilkannya ke layar monitor client. Sumbernya tetap berada di server dan di sana ia tidak berubah sama sekali. Inilah web yang “biasa-biasa” saja, web yang statis.

Orang kemudian mulai membuat metode baru yang membuat *server* harus melakukan lebih dari sekedar mengirim file. Jika permintaan dari *browser* mengarah ke suatu file program CGI (misalnya Perl, yang biasanya berekstensi *.pl), maka *server* mendeteksinya sebagai suatu permintaan untuk menjalankan program diluar *server. Server* lalu menjalankan program aplikasi yang simaksud. Program aplikasi lalu mengeluarkan hasil kerjanya ke *server*, kemudian mengirimkannya kembali ke *browser* dalam bentuk HTML seperti biasa. Bagi *browser* web, informasi yang diterima tetap serupa dengan dokumen HTML statis biasa, namun dokumen tersebut sudah bukan lagi sebuah salinan dari file yang ada di disk *server*, melainkan suatu informasi yang dihasilkan secara *on-the-fly* oleh program aplikasi. Informasi dari program dimasukkan ke dalam dokumen HTML sebelum dikirimkan ke browser. Metode ini sering disebut SSI *(Server Side includes).*



Gambar 2.2 Common Gateway Interface

Pendekatan cara CGI ini misalnya dengan script Perl, membutuhkan suatu file *template*, yaitu suatu file dokumen yang menjadi tempat penampung informasi hasil kerja program sebelum dikirimkan ke *browser* web. File ini berisi kode-kode khusus yang kemudian akan diganti dengan informasi hasil kerja penerjemah script CGI. Hasilnya, dokumen yang dikirim ke *browser* web sebenarnya adalah kombinasi dari informasi dinamis dari program aplikasi serta informasi statis dari file *template* tadi. Developer harus membuat dua file terpisah, yaitu script program dan file *template*nya.

2.7.3 Pendekatan Cara PHP

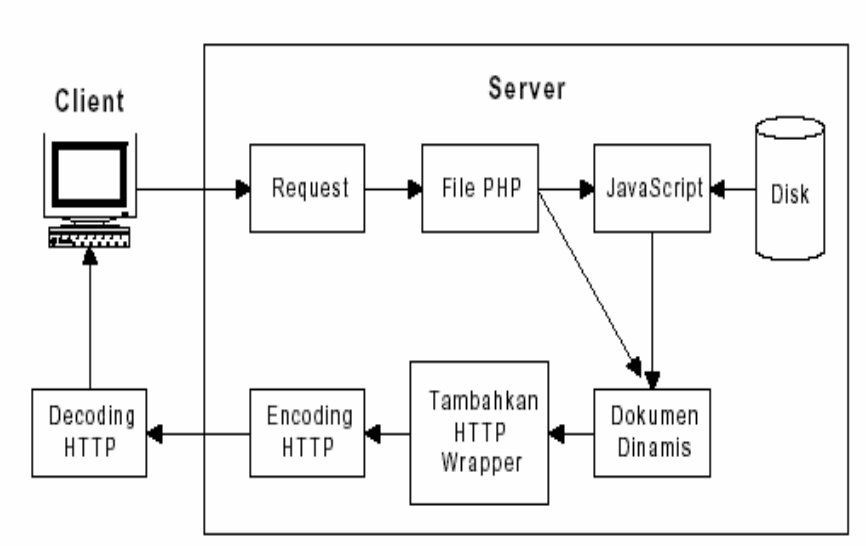
PHP menawarkan solusi yan lebih luwes. Dengan PHP, developer tidak perlu lagi berurusan dengan dua buah file terpisah seperti itu. *Browser* web mengcu secara langsung ke file yang dituju, yang lalu dibaca oleh *server*

sebagaimana file HTML statis biasa. Bedanya, sebelum dikirim balik ke *browser* web, *server* web memeriksa isi file dan menentukan apakah ada kode didalam file tersebut yang harus dieksekusi. Bila ada, kode-kode tersebut akan dieksekusi. Hasilnya dimasukkan ke dalam dokumen yang sama. *Server* web bekerja secara langsung terhadap file yang bersangkutan, tidak memanggil script terpisah seperti pada metode CGI. Seluruh kode dieksekusi di *server* (oleh karena itu disebut *server-side-script*).

PHP membuat proses pengembangan aplikasi menjadi mudah karena kelebihan-kelebihannya, yaitu :

1. Script (kode program) terintegrasi dengan file HTML, sehingga developer bisa berkonsentrasi langsung pada penampilan dokumen webnya.
2. Tidak ada proses compiling dan linking.
3. Berorientasi obyek (Object Oriented)
4. Sintaksis pemrogramannya mudah dipelajari, sangat menyerupai C dan Perl.
5. Integrasi yang sangat luas ke berbagai *server* database. Menulis web yang terhubung ke data base menjadi sangat sederhana. Database yang didukung oleh PHP : Oracle, Sybase, mSQL, MySQL, Solid, ODBC, PostgresSQL, Adabas D, FilePro, Velocis, Informix, dBase, UNIX dbm.

   PHP tidak terbatas untuk hanya menghasilkan keluaran HTML. Ia juga digunakan untuk menghasilkan gambar GIF, atau bahkan sumber gambar GIF yang dinamis.



Gambar 2.3 Prinsip Kerja PHP dan Apache

2.8 Squid Proxy Server

Squid web-cache proxy server adalah software proxy server yang bersifat open source yang didesain untuk berjalan di sistem UNIX dan keluarganya (tentu saja termasuk Linux). Squid tidak hanya dapat meng-*cache* objek-objek web saja, namun juga dapat meng-*cache* DNS dan network lookup lainnya. Meskipun pada awalnya didesain untuk sistem UNIX, namun Squid dapat pula diport ke Windows NT, namanya menjadi SquidNT. Info lebih lengkap mengenai squidNT, ada di URL berikut http://www.acmeconsulting.it/SquidNT/.

2.8.1 Squid dan Sejarahnya

Squid dikenal sebagai aplikasi cache yang handal. Squid berlisensi GPL (*GNU Public License*) atau *open source* yang dalam pembuatannya yang melibatkan banyak programmer. Dengan *open source* ini, squid bisa dikatakan

selalu dalam tahap pembuatan (*beta/develop version*) selama hidupnya (tidak hanya squid, hampir sebagian besar produk *open source* bersifat demikian). Namun tidak berarti squid jelek dan tidak stabil, melainkan kebalikannya.

Setiap orang yang mencoba produk *open source* otomatis menjadi *beta tester* yang diharapkan akan memberikan kontribusi berupa masukan kepada tim developer tentang *bug* dan kesalahan yang ditemuinya. Dalam perkembangannya tidak hanya segelintir orang yang menjadi *beta tester*, semakin menarik dan populer sebuah produk *open source*, semakin banyak *beta tester* yang bekerja secara sukarela. Oleh sebab itu proses ini menjadikan lebih banyak ditemukannya *bug* baru sekaligus dengan *bug fix*-nya. Pada akhirnya mengakibatkan membaiknya produk open source tersebut. Jadi, inilah yang menjadikan produk *open source* bias diandalkan. Demikian *life cycle* secara umum dari produk *open source.*

Squid dalam sejarah pembuatannya merupakan produk turunan dari Harvest Cache (dibuat oleh Harvest Project, http://harvest.cs.colorado.edu). Berdasarkan dari sumber tersebut, squid dikatakan sebagai server proxy cache yang mempunyai performansi tinggi dan mendukung operasional FTP, Gopher dan HTTP. Dalam memenuhi permintaan client, squid menangani dalam satu proses I/O (Input/Output) dan tidak bersifat blocking.

Squid bersifat monolitik dimana seluruh perkerjaan mulai dari menangani permintaan client, mengambil, mencari dan menyimpan objek serta pengaturan memori, seluruhnya itu hanya dilakukan oleh satu proses. Hampir tidak ada proses

lain, untuk proses I/O squid menyerahkan operasionalnya pada aplikasi lain yaitu *diskd*.

Squid dibuat oleh sebuah komunitas internet dan dipimpin oleh Duane Wessel dari *National Laboratory for Applied Network Research* yang dibiayai oleh *National Science Foundation*. List dari kontributor dapat dilihat dari http://www.squid-cache.org/CONTRIBUTORS. Squid berjalan diatas Unix dan variannya, juga termasuk Linux. Jenis varian Unix yang sampai saat ini diketahui telah berhasil menjalankan Squid diantaranya :

- Linux (semua distribusi)
- FreeBSD
- NetBSD
- OpenBSD
- BSDI
- Mac OS X
- OSF/Digital UNIX/Tru64
- IRIX
- SunOS/Solaris
- NeXTStep
- SCO UNIX
- AIX
- HP-UX

Beberapa jenis proxy server yang sering digunakan:

- Squid Web-cache: merupakan proxy server open source dan didesain untuk berjalan di sistem UNIX dan keluarganya.
- WinProxy: proxy server berbasis windows yang sifatnya free.
- WinGate Proxy Server: proxy server berbasis windows yang bersifat komersial.
- Microsoft Proxy: proxy server buatan Microsoft Corporation. Tentu saja sangat komersial.
- Dan lain-lain.

2.8.2 Cache dan Cache Server

Saat kita membuka browser dan menggetikkan URL http://www.google.com. Content yang diminta pada URL tersebut dinamakan "*Internet Object*" atau disingkat *object*. Pertama ia akan bertanya lebih dulu ke sebuah DNS (*Domain Name Server*). DNS mencari IP Address dari www.google.com dari databasenya dan memberikan jawaban kepada browser tadi. Setelah browser mendapatkan IP Address, maka ia membuka hubungan http ke web server tujuan. Web server mendengar adanya permintaan dari browser lalu memberikan content yang diminta. Setelah browser menerima content dan hubungan dengan web server bisa diputus. Content lalu ditampilkan dan disimpan dalam harddisk.

Content yang disimpan di harddisk biasanya disebut *cache object* yang nantinya digunakan jika pengguna kembali menggujungi site yang sama, misalnya dengan mengklik tombol back atau melihat history. Dalam kunjungan berikutnya,

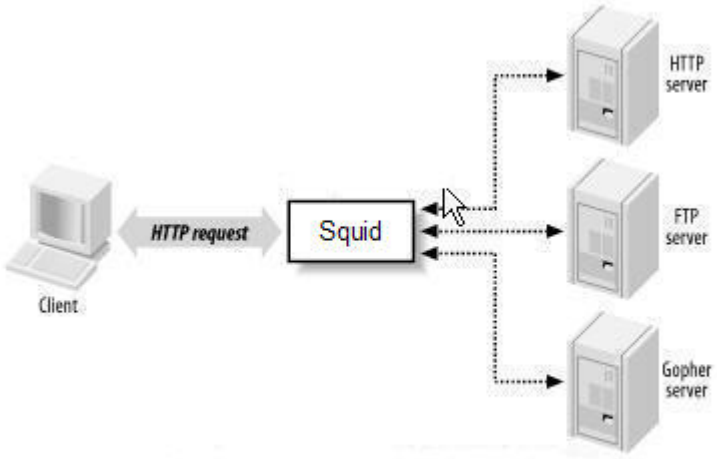
browser akan memeriksa validasi content yang disimpannya. Validasi ini dilakukan dengan membandingkan header content yang ada pada cache object dengan yang ada di web server. Jika content belum “basi” atau expired, cache object tadi yang akan ditampilkan browser. Semua browser mempunyai fitur cache seperti ini. Internet Explorer (IE) versi 5x mempunyai fitur baru mengenai caching ini. Jika IE mengunjungi site yang sama berulang-ulang, maka IE tidak akan mendownload ulang static image walaupun text pada site sudah berubah atau dengan kata lain IE lebih cerdas dalam mengatur validasi pada objek tertentu saja.

Cache object yang tersimpan dalam harddisk lokal ini hanya bisa dipakai oleh pengguna sendirian, tidak bisa dibagi kepada pengguna lainnya. Jika cache object tadi disimpan dalam sebuah server, dimana semua komputer terhubung dengan server tersebut, cache object bisa dipakai bersama-sama. Server ini dinamakan *Cache Server*.

Cache server merupakan salah satu dari jenis proxy server. Melihat dari namanya, cache arti umumnya adalah tempat penyimpanan sementara. Dalam kenyataanya, cache server berfungsi sebagai alat dimana “*internet object*” disimpan.

Cache server diletakkan pada titik di antara client dan web server. Pada contoh diatas, client akan meminta content www.google.com ke cache server, tidak langsung ke web server tujuan. Cache server bertanggung jawab untuk mendownload content yang diminta dan memberikannya kepada client. Content tadi disimpan pada hardisk local cache server. Lain waktu, ada client meminta

content yang sama, maka cache server tidak perlu mendownload ulang content ke web server tujuan tapi tinggal memberikan content yang sudah ada.



Gambar 2.4 Kedudukan Squid diantara Client dan Server

Content yang tersimpan dinamakan selanjutnya *object*. Dalam penyimpannanya disebut sebuah *list object* yang memuat informasi jenis objek seperti gambar dan html juga tidak lupa URL dari object yang bersangkutan. Object ini disimpan pada interval waktu tertentu.

Ada dua jenis metode caching, yaitu *aktif* dan *pasif*. Seperti telah diketahui, object yang disimpan bisa saja telah mencapai expired. Untuk memeriksanya dilakukan validasi. Jika validasi ini dilakukan setelah ada permintaan dari client, metode ini disebut *pasif*. Dan sebaliknya pada caching *aktif*, cache server mengamati object dan pola perubahannya. Misalkan pada sebuah object didapati setiap hari berubah setiap jam 12 siang dan pengguna biasa membacanya jam 14. Cache server tanpa diminta client akan memperbaharui objectnya pada jam 13. Dengan cara update otomatis seperti ini waktu yang dibutuhkan pengguna untuk mendownload content akan semakin sedikit.

Pada kondisi tertentu, kapasitas penyimpanan akan terkuras habis oleh object. Namun cache server mempunyai beberapa metode penghapusan untuk menjaga kapasitas tetap terjaga, sesuai dengan konfigurasi yang telah ditetapkan. Penghapusan ini didasarkan pada umur dan kepopuleran. Semakin tua umur object akan tinggi prioritasnya untuk dihapus. Dan juga untuk object yang tidak populer akan lebih cepat dihapus juga.

2.8.3 Object Cache

Pengaturan object merupakan salah satu bagian penting, mengingat object adalah benda yang selalu ‘diminta’, ‘dilempar’ dan ‘dibuang’ setiap waktu. Object disimpan pada dua level *cache_dir* yang besar levelnya didefinisikan pada file *squid.conf*. Object itu sendiri berisikan content URL yang diminta client dan disimpan dalam bentuk file binary. Masing-masing object mempunyai metadata yang sebagian dari isinya disimpan didalam memori untuk memudahkan melacak dimana letak object dan apa isinya. Pelacakan ini perlu cepat dilakukan dan oleh karena itu diletakkan didalam memori, sedangkan isi keseluruhan metadatanya sendiri ada di dalam hardisk.

2.8.4 Umur Object

Umur object merupakan sebuah ukuran waktu yang dihabiskan sebuah object untuk tinggal didalam hardisk cache. Umur object dibatasi oleh beberapa factor, yaitu :

- Metode penghapusan object

Object dihapus bisa dengan beberapa algoritma penghapusan.

- Kapasitas hardisk cache

  Semakin besar kapasitas cache, semakin lama umur object. Jika pemakaian hardisk sudah mendekati batas atas, penghapusan object akan semakin sering dilakukan.

- Banyaknya permintaan client

  Permintaan client dipenuhi dengan memberikan object yang diminta olehnya. Object ini akan disimpan dalam hardisk, jika banyk permaintaan maka banyak pula object yang harus disimpan.

2.8.5 Metode Penghapusan Object

Penghapusan object dari cache didasari oleh beberapa aturan (*policy*) yang terangkum dalam table berikut ini:

Tabel 1. Metode Penghapusan Object

Policy	Kriteria
Logistic Regression (LR)	Menghapus object dengan kemungkinan LR terkecil. Kemungkinan LR secara harafiah bisa diartikan sebagai besarnya kemungkinan object tersebut akan diakses di waktu yang akan datang.
Least Recently Used	Menghapus object berdasatkan waktu kapan object tersebut terakhir diakses. Semakin lama (besar) waktunya,

(LRU)	kemungkinan untuk dihapus semakin besar.
Least Frequently Used (LFU)	Menghapus object berdasarkan dari object yang paling jarang diakses.
First In First Out (FIFO)	Menghapus object berdasarkan waktu masuk kedalam cache. Object yang masuk paling awal akan dihapus pada prioritas tertinggi.
Random	Menghapus object secara acak.

2.8.6 Intepretasi File Log

File log yang dihasilkan squid ada tiga file yaitu : *access.log*, *store.log*, dan *cache.log.* Access.conf merupakan catatan aktifitas akses user seperti URL, user name, bytes dan lain-lain. *Store.log* adalah catatan penyimpanan object ke dalam hardisk. Sedangkan *cache.log* adalah pesan yang dibuat squid sewaktu running, biasanya berupa pesan error.

File log ini sangat penting untuk dianalisa dan dicermati jika terjadi sesuatu kesalahan pada squid. Biasanya pada file cache.log mencatat kejanggalan mulai dari kehilangan parent sampai disk full. Untuk access.log banyak terdapat tool untuk membaca dan mengolahnya lebih lanjut menjadi statistic grafis. Tool seperti ini contohnya *prostat* dan *calamaris.* Dari hasil outputnya bisa diperoleh informasi kemana saja user *surfing,* TOP 10 URL yang paling sering dikunjungi dan berapa besar bytes yang digali dari hardisk server.

2.8.7 Kode Keberhasilan Squid

Pada kolom code atau *result* pada file access.log secara garis besar terbagi atas dua bagian. Yaitu TCP dan UDP.

- TCP_HIT

  Object yang diminta ada di hardisk cache

- TCP_MISS

  Object yang diminta tidak ada di hardisk cache

- TCP_REFRESH_HIT

  Object ada tetapi dalam keadaan STALE. Stale ini berarti object sudah agak lama umurnya. Query IMS menghasilkan pesan "*304 not modified*". Query IMS ini dapat diartikan sebagai permintaan client yang menginginkan content yang terbaru dari sebuah situs.

- TCP_REF_FAIL_HIT

  IMS gagal menghasilkan object yang baru, maka object yang lama akan diberikan. Ini biasanya pada keadaan cache tidak dapat mengambil object terbaru karena koneksi sedang terputus dan object yang diminta ada didalam cache tetapi umurnya sudah lama.

- TCP_REFRESH_MISS

  Sama dengan TCP_MISS_HIT, perbedaannya hanya anatara HIT dan MISS.

- TCP_CLIENT_REFRESH_MISS

  Object tidak ada karena memang disebutkan "no_cache" dalam file squid.conf, oleh karena itu harus diambil ulang.

- TCP_SWAPFAIL_MISS

  Object ada di cache tetapi tidak bias diakses.

- TCP_NEGATIVE_HIT

  Object yang diminta ada namun berupa object negative. Object negative misalnya halaman yang tidak ada pada server original (404 nnot found). Jadi, cache akan memberikan keterangan “404 not found” kepada client.

- TCP_MEM_HIT

  Object yang diminta ada dan terdapat dimemori, jadi tidak perlu mengambil dari hardisk. Object ini biasanya disebut “hot object”.

- TCP_DENIED

  Akses ditolak. Ini disebabkan oleh http_access deny dimana client termasuk dalam acl yang ditolak.

- TCP_OFFLINE_HIT

  Object ada dan cache server sedang bekerja pada mode offline. Dalam mode ini cache server tidak akan melihat validasi object (expired atau tidak).

- UDP_HIT

  Object ada di dalam cache (melalui protocol UDP).

- UDP_MISS

  Object tidak ada di dalam cache.

- UDP_DENIED

  Akses UDP ditolak.

- UDP_INVALID

  Permintaan UDP tidak valid atau salah sintaks.

- NONE

  Pengembalian pesan error kepada client atau bias juga permintaan cachemgr.

- UDP_HIT_OBJ

  Object baru saja HIT oleh object lain, tetapi saat diminta kembali object sudah tidak ada (sudah dihapus).

2.8.8 Jenis-jenis Cache

Masalah lain yang dihadapi dalam penduplikasian object adalah bagaimana menjaga agar object yang diduplikasikan sama dengan object yang terakhir ada di server asli. Server mirror atau cache harus melakukan pembaruan kembali object agar *user* percaya bahwa object yang ia terima dari server mirror atau server proxy-cache sama dengan object yang terdapat di server. Saat ini terdapat setidaknya dua jenis pembaruan kembali yang biasa digunakan yaitu jenis *persistent* dan jenis *non-persistent*. Pada jenis *persistent*, jika terdapat object dalam cache maka object yang disampaikan ke *user* adalah object yang berada dalam cache tersebut. Object tersebut selalu diambil dari cache kecuali jika *user* memaksa agar object diambil langsung dari sumbernya. Contoh jenis ini adalah seperti yang digunakan oleh *browser* Netscape. Untuk memerintahkan Netscape agar mengambil langsung dari sumber object, *user* harus memilih ikon *reload*. Kelemahan utama jenis ini adalah pembaruan kembali object harus dipaksakan

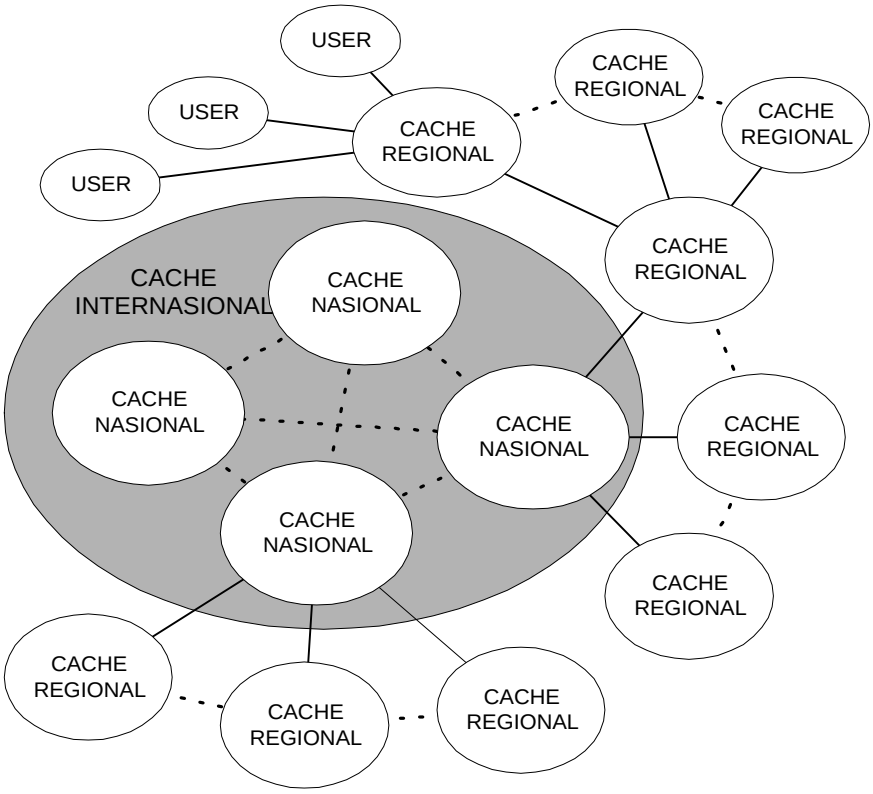
dan tidak ada mekanisme untuk memeriksa apakah object yang disimpan masih sama dengan object yang terdapat pada server asal atau object harus diambil kembali dari server asal tersebut.

Berbeda dengan cache jenis *persistent*, cache *non-persistent* memiliki mekanisme untuk memperbarui object di dalam cache sehingga object yang diterima oleh *user* dapat selalu sama dengan object di server asal tanpa perlu dipaksa oleh *user*. Mekanisme pembaruan kembali object dalam cache ini menggunakan dua buah algoritma yang saling mendukung: algoritma *Ageing* dan algoritma *Least Recently Used* (LRU). Algoritma LRU bertugas untuk menjaga agar selalu tersedia tempat untuk menyimpan object yang baru diakses. Jika cache sudah hampir penuh, algoritma ini menghapus object-object yang lama tidak diakses *(least recently used*) sampai batas tertentu. Algoritma *Ageing* memeriksa tanggal object yang terdapat di dalam cache untuk menentukan apakah object tersebut harus dihapus. Di samping itu algoritma *Ageing* juga menentukan apakah object yang sedang diakses *user* perlu diambil dari server asal atau cukup dengan menggunakan object yang terdapat di dalam cache. Server proxy umumnya menggunakan cache *non-persistent* karena kelebihan yang disebutkan di atas.

2.8.9 Hirarki Cache

Antara cache server bisa terjalin saling kerjasama. Protocol yang menghubungkan antara server cache ini disebut *Internet Cache Protocol* (ICP). Dengan ICP, system cache bias mempunyai hirarki. Hirarki dibentuk oleh dua jenis hubungan, yaitu parent dan sibling.

- Parent adalah cache server yang wajib mencarikan content yang diminta oleh client.
- Sibling adalah cache server yang wajib memberikan content yang diminta jika memang tersedia. Jika tidak, sibling tidak wajib untuk mencarikannya.

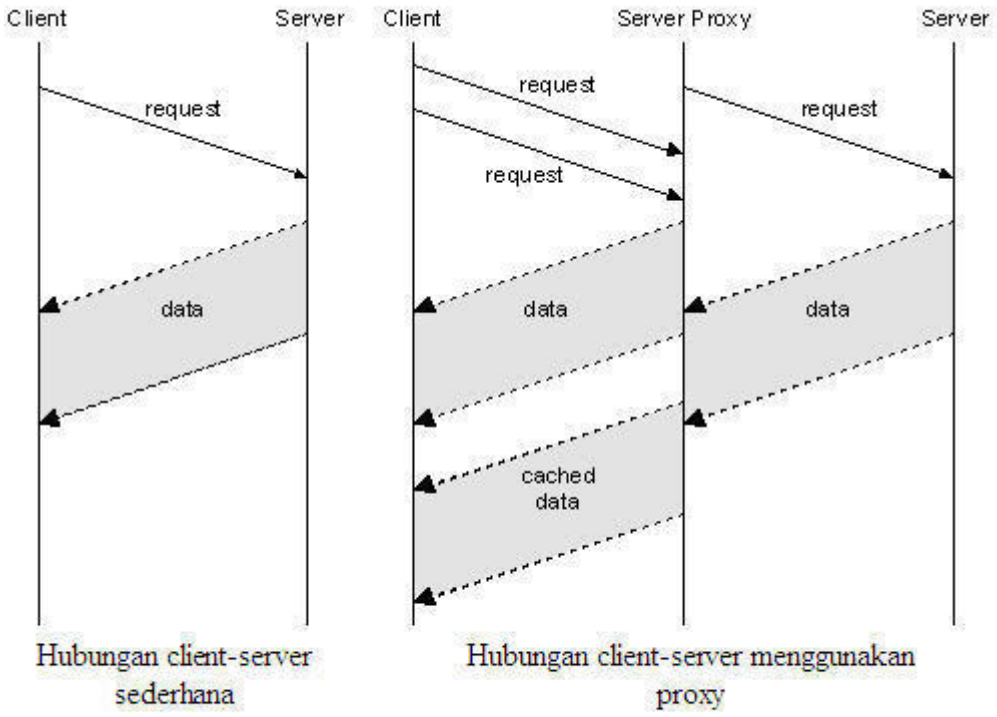


Gambar 2.5 Prototipe hierarki cache

2.8.10 Hubungan Client dan Squid

Dari sisi client, aplikasi browser yang telah dikonfigurasi meminta layanan http pada port 80 kepada cache server dengan port yang telah disesuaikan dengan milik server. Namun, dari sisi server squid nomor port yang dipakai bukan lagi 80, tetapi akan diredirect ke port 8080 atau 3128 sesuai dengan konfigurasi server squid.

Pada saat browser mengirimkan header permintaan, sinyal http request dikirimkan ke server. Header tersebut diterima squid dan dibaca. Dari hasil pembacaan, squid akan memparsing URL yang dibutuhkan. Lalu URL ini dicocokkan dengan database cache yang ada. Database ini berupa kumpulan metadata (semacam header) dari object yang sudah ada didalam hardisk. Jika ada, object akan langsung dikirim ke client dan tercatat dalam logging bahwa client telah mendapat object yang diminta. Dalam log kejadian ini dicatat sebagai TCP_HIT. Sebaliknya jika object yang diminta tidak ada, squid akan memintanya dari peer (jika ada) atau langsung ke server tujuan. Setelah mendapatkan objectnya, squid akan menyimpan object tersebut ke dalam hardisk. Selama dalam proses download, object ini dinamakan *object in transit* yang sementara menghuni ruang memori. Dalam masa download tadi, object mulai dikirimkan ke client dan setelah selesai, kejadian ini tercatat dalam log sebagai TCP_MISS.



Gambar 2.6 Model Hubungan Client Serve

2.8.11 Hubungan Squid dengan Peer

Peer ada dua jenis, yaitu parent dan sibling. Sibling boleh dibilang cache server berada satu level dibawah parent. Dua jenis peer inilah yang nantinya saling bergandengan membentuk jaringan hirarki cache.

Internet Cache Protocol (ICP) sebagai protocol cache berperan dalam menanyakan ketersediaan object dalam cache. Dalam sebuah jaringan cache yang mempunyai sibling, akan mencoba mencari object yang dibutuhkan dari sibling dan bukan kepada parent. Cache akan mengirimkan ICP kepada sibling, dan sibling akan membalasnya dengan informasi ketersediaan, ada atau tidak object yang diminta. Bila ada, cache akan mencatatkan ICP_HIT dalam lognya. Setelah kepastian object bias diambil dari sibling, lalu cache akan mengirimkan sinyal http ke sibling untuk mengambil object yang dimaksud. Dan setelah

mendapatkannya, cache akan mencatat SIBLING_HIT. Jika ternyata sibling tidak menyediakan object yang diminta, cache akan memintanya kepada parent. Sebagai parent ia wajib mencarikan object tersebut walaupun ia sendiri tidak mempunyainya (TCP_MISS). Setelah object didapatkan dari server original, object akan dikirimkan ke cache child (cache yang mengambil object dari parent) tadi. Setelah mendapatkannya, cache child akan mencatatkan log PARENT_HIT.

2.8.12 Kerugian dan Keuntungan Proxy Server

1. Keuntungan menggunakan proxy server:
    - Dapat menghemat biaya bandwidth.
    - Mempercepat koneksi karena file-file web yang direquest (selanjutnya disebut *object*) disimpan di dalam cache sehingga tidak perlu keluar menuju internet.
    - Dapat mengatur kecepatan bandwidth untuk subnet yang berbeda-beda.
    - Dapat melakukan pembatasan untuk file-file tertentu.
    - Dapat melakukan pembatasan akses kepada situs-situs tertentu (misalnya situs porno).
    - Dapat melakukan pembatasan download untuk file-file tertentu (misalnya file-file mp3, wav, dsb).
    - Dapat melakukan pembatasan waktu-waktu yang diperbolehkan untuk download.

- Dapat melakukan pembatasan siapa saja yang boleh mengakses internet dengan menggunakan autentikasi. Autentikasi yang biasa digunakan bisa basic, digest, ataupun ntlm.
- Dapat melakukan pembatasan-pembatasan lainnya.

2. Kerugian menggunakan proxy server:
    - Pintu keluar menuju gerbang internet hanya lewat proxy, sehingga ketika terjadi overload, akses internet menjadi lambat.
    - User akan melihat file yang kadaluarsa jika cache expire time-nya terlalu lama, sehingga meskipun di website file tersebut sudah berubah, user masih melihat file yang tersimpan di cache memory.
    - Karena koneksi internet harus melalui gerbang proxy terlebih dahulu, maka kecepatan akses bisa jadi lebih lambat daripada kita melakukan koneksi langsung. Dalam hal ini keduanya akan mengakses file internet secara langsung.