PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK SERVER PROXY BERDASARKAN CONTENT CACHE

TUGAS AKHIR

Oleh :

PUJI WIDI PRAYITNO
NRP. 5196 100 067

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2003

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK SERVER PROXY BERDASARKAN CONTENT CACHE

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui,
Dosen Pembimbing

Febrilian Samopa, S.Kom, M.Kom
NIP. 132 206 856

SURABAYA,
2003

ABSTRAK

Internet sebagai sunber daya informasi mempunyai peranan yang begitu penting bagi masyarakat modern. Saat ini hamper setiap lapisan masyarakat telah mengenal Internet , dan berinteraksi dengannya sesuai dengan kepentingan mereka masing-masing

Dalam Tugas Akhir ini dikembangkan sebuah aplikasi server proxy yang dapat merequest link-link yang terdapat pada halaman yang telah direquest oleh user dalam sebuah jaringan Internet (content cache).

Keuntungan yang diperoleh dari penggunaan aplikasi ini diukur berdasarkan selisih waktu yang ditempuh antara sebelum menggunakan aplikasi ini dan setelah mempergunakannya

KATA PENGANTAR

Alhamdulillah, puji syukur kepada Allah Subhanahu Wa Ta'ala yang karena kehendakNya saya berhasil mengerjakan Tugas Akhir ini. Meskipun mengalami banyak hambatan dalam masa perkuliahan dan khususnya pada pengerjaan Tugas Akhir, namun saya memperoleh hal-hal baru yang memperkaya pemahaman saya pada bidang komputasi.

Dalam Tugas Akhir ini saya mengupas beberapa aspek dari squid sebagai sebuah server proxy yang freeware dan apa-apa yang saya lakukan untuk membuatnya menjadi sebuah software proxy yang content cache.

Saya mengucapkan terimakasih kepada semua pihak yang membantu saya dalam pembuatan Tugas Akhir baik secara langsung maupun tidak langsung. Tidak semuanya dapat saya sebutkan satu persatu, akan tetapi hal ini bukan berarti menafikkan jasa yang telah diberikan, karena saya tidak berarti apa-apa tanpa bantuan orang lain. Pihak-pihak tersebut antara lain adalah :

1. Ayah dan Ibu, yang memelihara saya sejak kecil dengan kasih sayang. Ibu, yang sangat kurindukan, semoga engkau bahagia di sana, dan diringankan penderitaanmu, amin
2. Semua kakak, yang telah menjadi pendorong semangat saya untuk segera lulus dari kampus ini
3. Bapak Febriliyan Samopa, S.Kom, M.Kom, sebagai dosen wali dan dosen pembimbing Tugas Akhir, terimakasih atas segala fasilitas, nasihat dan saran yang diberikan.

4. Bapak Rully Soelaiman, S.Kom, M.Kom, atas saran-saran dan masukannya
5. Bapak Yudhi Purwananto S.Kom, M.Kom sebagai Kepala Jurusan Teknik Informatika
6. Ibu Nanik Suciati S.Kom, M.Kom sebagai Sekretaris Jurusan Teknik Informatika
7. Bapak Dr. Ir. Arif Djunaidy Ph.D sebagai Dekan Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember
8. Ibu Chastin Fatichah, yang banyak membantu dalam penyelenggaraan Tugas Akhir ini
9. Teman-teman Laboratorium Komputing, Antonius Anang, Suyudi, Irwan S. Kilay, Surateno, Myke N, Alif Rufiana, Dwi P, Tuti, Nissa, Mashadi S, Budi T.C, Sugeng Hariyono, Wiwik D.S.K, Anita Hidayati, Cahyono, Eko Pramono, Franky S, Nafiri F, Gershom Michael, Eben, Victoria S, dll
10. Teman-teman yang membantu secara teknis maupun non teknis, Kamas 00, Nita 98, Eggy 00, Didit R.H
11. Bapak Imam Kuswardayan S.Kom
12. Bapak Anung Yulianto
13. Semua staf tata usaha Teknik Informatika
14. Teman-teman Blok U-ITS, Darlis, Ali, Andik, Nugroho, Ajie
15. Komunitas hacker server proxy squid yang telah mencurahkan pikirannya guna perbaikan program ini. Terimakasih kepada mereka

yang tidak merasa rugi berbagi ilmu dan pengetahuannya untuk kebaikan bersama, tanpa memungut biaya apapun

Saya merasa bahagia dapat menyelesaikan penulisan buku ini. Dan saya berharap agar buku ini berguna bagi kehidupan dan pengembangan ilmu pengetahuan. Saya sebagai penulis dengan senang hati menerima segala kritik dan saran yang membangun.

Surabaya, April 2003

Puji Widi Prayitno

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1. LATAR BELAKANG

Internet sebagai sumber daya informasi mempunyai peranan yang begitu penting bagi masyarakat modern. Saat ini hampir setiap lapisan masyarakat telah mengenal Internet, dan berinteraksi dengannya sesuai dengan kepentingan mereka masing-masing.

Hal penting yang perlu dipertimbangkan dalam berinteraksi dengan Internet adalah faktor-faktor biaya dan kecepatan. Keduanya berkaitan dengan masalah efisiensi materi dan waktu. Meskipun telah banyak solusi yang ditawarkan untuk mengatasi hal-hal tersebut, namun usaha untuk mengoptimalkan faktor-faktor ini harus terus dilakukan.

Sekumpulan komputer dalam keadaan terhubung menjadi satu jaringan memiliki keuntungan jauh lebih besar daripada tidak terhubung dalam jaringan. Selain keuntungan berbagi sumber daya (penggunaan file secara bersama-sama), terdapat pula keuntungan ketika berkomunikasi dengan lingkungan luar (jaringan di luar), misalnya Internet. Koneksi ke Internet cukup hanya diwakili oleh sebuah komputer saja, yaitu komputer server.

Penggunaan server proxy cache (cache proxy server) merupakan salah satu solusi yang ditawarkan untuk meningkatkan efisiensi. Server proxy cache menyimpan data Internet yang melaluinya dalam cache pada proxy. Pada saatnya,

data-data tersebut diminta kembali bukan hanya oleh klien yang meminta (*request*) pertama kali, akan tetapi juga klien lain dalam jaringan komputer yang dibawahi oleh server proxy tersebut. Server proxy tidak perlu mengambil data dari server web aselinya, akan tetapi dari cache yang terdapat padanya.

Cara ini dapat menghemat waktu, karena file data yang didownload berasal dari jaringan lokal, bagi setiap klien dalam jaringan tersebut. Lalu lintas data client server menjadi pendek dan singkat, sehingga bandwidth yang disediakan dapat dipergunakan untuk kepentingan lainnya. Selain itu cara ini juga hemat biaya, karena jaringan ini tidak harus terhubung secara langsung ke server web (online).

Meskipun cara di atas sudah bekerja dengan baik, para pengembang perangkat lunak selalu mencari celah yang dapat dikembangkan agar menjadi produk yang kompeten. Server proxy di atas bekerja dengan cara pasif. Maksudnya, server hanya menerima dan menyimpan data-data yang melewatinya akibat request dari pengguna Internet.

Pengembangan server proxy menjadi sebuah alat yang aktif, perlu dilakukan. Membuat server proxy mampu secara proaktif merequest data dari Internet memiliki kemungkinan untuk dikembangkan. Dalam hal ini, penulis mencoba melakukan pengembangan pada proxy server agar dapat bekerja aktif.

1.2. TUJUAN

Membuat fitur baru pada perangkat lunak server proxy sehingga memiliki kemampuan secara otomatis melakukan request terhadap server halaman-halaman web yang berhubungan dengan halaman web yang sedang diminta oleh user.

1.2. PERMASALAHAN DAN BATASAN MASALAH

Permasalahan pada pengerjaan Tugas Akhir ini adalah bagaimana menjadikan software server proxy yang sudah ada saat ini agar memiliki kemampuan secara aktif melakukan permintaan (fetching) halaman-halaman yang berhubungan dengan halaman yang telah difetch sebelumnya dan menyimpannya dalam cache

Untuk mengerjakannya, penulis melakukan beberapa pembatasan permasalahan sebagai berikut :

1. Server Proxy yang dipilih menjadi objek penelitian dan percobaan adalah **squid– 2.4.STABLE6**
2. Data Internet yang di load/fetch adalah file HTML
3. Titik berat tugas akhir ini adalah membuat server proxy mampu mengambil data yang link-nya terdapat dalam halaman web.

batasan tersebut diatas dibuat untuk lebih memusatkan perhatian penulis pada lingkungan yang jelas.

1.3. METODOLOGI

Metodologi yang dipergunakan dalam mengerjakan Tugas Akhir ini meliputi

1. Studi Literatur dan Penelitian

   Jumlah referensi yang sangat terbatas dari software squid mendorong penulis mempelajari kode sumber squid lebih detil. Literatur yang diperoleh berasal dari *mailing list* squid.

   Selain itu penulis juga mempelajari buku-buku lain yang mendukung pengerjaan tugas akhir ini, seperti tentang masalah Internet, pemrograman C, sistem operasi Linux, dll.

2. Perancangan

   Pada tahap ini dilakukan perancangan konsep, perancangan struktur data dan perancangan antar muka guna diterapkan pada perangkat lunak yang hendak dibuat.

3. Pembuatan

   Pembuatan perangkat lunak mengikuti garis besar yang telah direncanakan pada proses perancangan.

4. Ujicoba

   Tahap uji coba dan tahap pembuatan saling melengkapi. Pada kedua tahap ini, masing-masing proses dapat bergantian secara berulang-ulang. Hal ini dilakukan untuk mendapatkan hasil yang sesuai dengan harapan.

5. Penyusunan Buku Tugas Akhir

   Penyusunan buku tugas akhir merupakan hasil dari keseluruhan tahap yang telah dilakukan sebelumnya.

1.4. SISTEMATIKA PEMBAHASAN

Agar dalam pembahasan Tugas Akhir ini lebih sistematis dan terarah, maka pembahasan dibagi menjadi lima bab dengan sistematika sebagai berikut :

1. BAB I PENDAHULUAN, berisi Latar Belakang, Permasalahan, Pembatasan Masalah, Metodologi, Sistematika, dan Tujuan.dari Tugas Akhir yang disusun.
2. BAB II DASAR TEORI, berisikan teori penunjang antara lain pengetahuan mengenai Internet, proxy, cache, dan pemrograman socket.
3. BAB III SQUID SEBAGAI PROXY SERVER, menerangkan secara detil squid sebagai server proxy.
4. BAB IV PERANCANGAN DAN IMPLEMENTASI, membahas solusi yang diberikan oleh penulis dalam bentuk perancangan dan implementasi perangkat lunak.
5. BAB V UJI COBA, melakukan pengujian terhadap program yang telah dibuat dan penghitungan hasil berdasarkan parameter yang dimasukkan.
6. BAB VI SARAN DAN PENUTUP, memberikan kesimpulan dan saran dari Tugas Akhir ini dan merupakan akhir dari penulisan Tugas Akhir ini.
7. Lampiran, merupakan pelengkap dari apa-apa yang telah dibahas dalam bab-bab sebelumnya, antara lain berisi kode sumber dan gambar.

BAB II

DASAR TEORI

BAB II

DASAR TEORI

2.1 INTERNET

ARPANET merupakan proyek pengembangan jaringan komputer yang dilakukan oleh Departemen Pertahanan Amerika Serikat untuk keperluan keamanan. Jaringan ini menghubungkan komputer dari berbagai macam vendor dan protokol yang berbeda-beda bersama-sama dengan jaringan komputer lainnya membentuk jaringan komputer yang lebih besar.

Seiring dengan perkembangan waktu, *ARPANET* dipergunakan tidak hanya dalam lingkungan militer, namun juga masuk ke kawasan universitas, perusahaan dan komunitas pemakai.

Prosedur dibuat untuk mengatur penempatan alamat dan membuat standard bagi network. Seiring dengan semakin berkembangnya Local Area Network, banyak host yang menjadi *gateway* bagi local network. Sebuah layer / lapisan network untuk melakukan operasi antar network ini dikembangkan dan dinamakan Internet Protocol (IP). Selanjutnya grup-grup network terbentuk berdasarkan IP ini. Jaringan-jaringan saling berinteraksi dalam operasi karena IP. Kumpulan dari jaringan yang saling beroperasi ini dinamakan Internet. Istilah Internet dimaksudkan sebagai kumpulan jaringan komputer (network) dari jaringan komputer (subnetwork) [EKR-89]. Dengan kesamaan umum yang

dimiliki masing-masing subnetwork ini adalah penggunaan TCP / IP sebagai protokol komunikasi.

Organisasi Internet dikontrol oleh *Internet Advisory Board* (*IAB*). Diantara hal yang ditangani oleh IAB adalah Internet *Engineering Task Force* (*IETF*) yang menangani aspek teknis dan penerapan yang berhubungan dengan *Internet dan Internet Research Task Force* (*IRTF*) yang melakukan penelitian.

Internet memiliki empat aplikasi utama yang populer, yaitu : surat elektronik (email), news, transfer file dan remote login yang masing-masing memiliki kegunaan sebagai berikut :

Email memiliki kemampuan menyusun, mengirim dan menerima surat elektronik dengan mempergunakan protokol *Simple Mail Transfer Protokol* (*SMTP)*

News merupakan forum khusus dimana sekelompok pengguna dengan kepentingan tertentu dapat saling berkomunikasi

Transfer File adalah kemampuan untuk mengirimkan file antar komputer (*host*) dalam satu jaringan komputer tanpa dipengaruhi oleh kendala jarak

Remote Login adalah kemampuan untuk melakukan *login* dari terminal yang berbeda dengan mengabaikan kendala jarak.

2.2 HTTP

Hypertext Transfer Protocol (HTTP) adalah sebuah protokol level aplikasi untuk sistem informasi hypermedia yang kolaboratif dan terdistribusi. Protokol ini adalah protokol yang generik dan stateless yang dapat dipergunakan untuk banyak hal diluar kegunaannya sebagai hypertext, seperti server name dan sistem

manajemen objek terdistribusi, melalui ekstensi metode request, kode error, dan header. Sebuah feature HTTP adalah penulisan dan negosiasi representasi data, memungkinkan sistem dibangun secara mandiri dari/terhadap data yang ditransfer [NWG-99].

HTTP mempergunakan protokol umum untuk komunikasi antara user agent dan proxy/*gateway* dengan sistem Internet lainnya, meliputi komunikasi yang disupport oleh SMTP, NNTP, FTP, Gopher dan Wais. Dalam hal ini HTTP memungkinkan hypermedia dasar untuk mengakses sumberdaya yang berasal dari aplikasi yang berbeda.

HTTP adalah protokol request / response, sebuah klien mengirimkan sebuah request ke server dalam bentuk metode request, URI, dan versi protokol, diikuti oleh message yang MIME-like, yang berisikan modifier request, informasi klien, dan isi bodi atas koneksi dengan sebuah server. Server merespon dengan sebuah status line, yang meliputi versi protokol dari message, dan sebuah kode sukses atau error, diikuti oleh message yang MIME-like, yang berisikan informasi server, metainformasi entity, dan isi entity-body. Dalam hal ini entity adalah informasi yang ditransfer sebagai request atau respon [NWG-99] .

Kebanyakan komunikasi HTTP dimulai dengan sebuah user agent dan terdiri dari sebuah request untuk ditrerapkan pada sumberdaya pada origin server. User agent adalah klien yang memulai sebuah request, misalnya browser, editor, spider (web-traversing robot), atau tool end user lainnya [NWG-99].

Komunikasi HTTP pada umumnya berjalan di atas koneksi TCP/IP. Port default dari TCP adalah 80, akan tetapi port lain dapat dipergunakan. Hal ini tidak

mencegah HTTP diterapkan di atas semua protokol lainnya pada Internet, atau pada network lainnya. HTTP hanya sebuah diasumsikan transport reliabel.

Uniform Resource Identifiers (URI) adalah string terformat yang mengidentifikasi, –dengan nama, lokasi, atau karakteristik lainnya – sebuah resource [NWG-99] . URI memiliki banyak sebutan, antara lain WWW address, Universal Document Identifier, dan Universal Resource Locator and Names (URN).

Request-Line dimulai dengan token method (metode), diikuti dengan request URI dan versi protokol, dan berakhir dengan CRLF. Elemen dipisahkan dengan karakter SP. Tidak ada CR atau LF kecuali di akhir sekwen CRLF.

```
Request-Line=Method SP Request-URI SP HTTP-Version
CRLF
```

Method mengindikasikan metode yang dilakukan pada resource yang diidentifikasi oleh Request-URI. Metode bersifat case sensitif.

```
Method      = "OPTIONS"
            | GET
            | HEAD
            | POST
            | PUT
            | DELETE
            | TRACE
            | CONNECT
            | Extension-method
Extension-method = token
```

Metode GET mengambil informasi yang diidentikasi oleh Request-URI.. Metode Head sama dengan metode GET, akan tetapi tidak mengembalikan message-body dalam responnya.

Metode POST dipergunakan untuk request dimana origin server menerima entity dalam request sebagai sebuah subordinat baru dari resource yang diidentifikasikan dengan Request-URI dalam Request-Line.

Metode PUT dipergunakan untuk request dimana entity yang disimpan berada dibawah Request-URI yang disuplai.

Metode DELETE dipergunakan untuk request dimana origin server menghapus resource yang diidentifikasikan oleh Request-URI.

Metode TRACE memungkinkan klien untuk melihat, apakah yang sedang diterima di akhir rantai request dan menggunakan data tersebut untuk memperoleh informasi secara testing atau diagnostik. Metode ini dipergunakan untuk memanggil loopback layer aplikasi dari message request.

Metode CONNECT dipergunakan untuk reserve nama. Metode ini dipergunakan dengan sebuah proxy yang dapat secara dinamis berganti menjadi sebuah tunnel. Tunnel adalah sebuah program yang bertindak sebagai relay diantara dua koneksi.

Daftar metode yang diperbolehkan oleh sebuah resource dapat dispesifikasikan dalam sebuah fields Allow header.

2.2 PROXY

Dalam suatu jaringan komputer, masing – masing komputer dapat berbagi sumber daya (misalnya data, program) satu sama lain. Namun biasanya terdapat satu buah komputer yang memang disediakan untuk keperluan penyimpanan data bagi semua anggota jaringan. Komputer yang menyediakan sumber daya ini dinamakan server. Dengan kata lain server adalah program aplikasi yang menerima koneksi untuk melayani request dengan mengirimkan respon kembali [NWG-99] . Sedangkan origin server adalah Server tempat resource berada atau dibuat [NWG-99] . Secara teknis klien adalah program yang membuat koneksi dengan tujuan untuk membuat request [NWG-99]

Proxy adalah program intermediate yang berlaku sebagai server dan klien sekaligus untuk membuat request pada sebagian klien yang lainnya. Request dilayani secara internal atau melewatkan ke server lainnya dengan translasi yang mungkin. Sebuah proxy harus memenuhi kebutuhan klien dan server ini [NWG-99].

Pengertian lain dari proxy adalah server yang memiliki kemampuan untuk melakukan aksi terhadap komputer lainnya yang tidak dapat dilakukannya sendiri [WSO-00], Pada umumnya server juga bekerja sebagai proxy pada sebuah jaringan komputer.

Proxy dapat dikategorikan berdasarkan kegunaannya. Setidaknya terdapat dua macam proxy berdasarkan kegunaan, yaitu : proxy firewall dan proxy cache.

Proxy firewall merupakan penyaring (filter) bagi jaringan yang di dibawahinya dan dipergunakan untuk mempertahankan keamanan jaringan dari

gangguan luar. Dengan proxy firewall, maka tidak semua data dapat keluar masuk jaringan. Proxy jenis ini tidak memiliki cache untuk menyimpan data.

Berbeda dengan firewall, proxy cache memiliki cache untuk menyimpan data yang telah dimintanya (fetch) dari jaringan luar. Cache ini dipergunakan untuk mengurangi kepadatan arus data keluar masuk jaringan, dan memperkecil kendala waktu yang dipergunakan untuk mendownload data dari origin server, apabila server tersebut terletak jauh dari klien yang memintanya. Dengan demikian cache proxy menekanakan efisiensi daripada keamanan.

2.3 CACHE

Cache adalah penyimpanan lokal data respon dan subsistem yang mengontrol media simpan data, pengambilan data dan penghapusan. Sebuah cache menyimpan respon yang cacheable untuk mengurangi waktu respon dan konsumsi bandwidth network di waktu berikutnya, untuk request yang sama [DWC-97]

Sebagaimana telah disinggung di atas, cache dipergunakan untuk meningkatkan efisiensi kinerja pada jaringan. Cache merupakan tempat (space) dalam media penyimpan yang dipergunakan untuk menyimpan secara sementara data-data yang telah didownload oleh jaringan untuk diambil kembali sewaktu-waktu diperlukan oleh anggota dari jaringan tersebut (klien).

Hal yang perlu dipertimbangkan dari cache ini adalah usia suatu data disimpan didalamnya (age). Age dari sebuah respon didefinisikan sebagai waktu sejak dikirim oleh server atau secara sukses divalidasi oleh server [DWC-97]. Data-data yang keluar masuk jaringan begitu banyak dan beraneka ragam. Dengan

kapasitas cache yang terbatas, tidaklah mungkin semua data tersebut disimpan dalam cache. Di sisi lain tidak semua data diperlukan kembali oleh klien.

Kadangkala data yang diberikan oleh server sengaja dilewatkan begitu saja melalui proxy tanpa harus melalui penyimpanan. Hal ini dilakukan untuk menjaga agar data yang diterima oleh klien merupakan data yang selalu baru.

Proxy Cache ketika menerima permintaan data dari klien akan memeriksa terlebih dahulu cache untuk mengetahui apakah data yang dimaksudkan terdapat atau tidak, apabila ada maka data tersebutkan disampaikan kepada klien yang meminta. Namun apabila data yang dimaksud tersebut tidak ada, maka proxy akan meneruskan ke proxy cache tetangga atau server sesungguhnya.

2.3.1 ICP, Protokol kontrol Internet

Internet Cache Protocol (ICP) adalah sebuah format data yang dipergunakan untuk komunikasi antar cache Web. Meskipun cache Web mempergunakan HTTP untuk transfer data obyek, cache mengambil keuntungan dari sebuah protokol komunikasi yang lebih sederhana dan ringan. ICP terutama dipergunakan dalam sebuah mesh cache untuk mencari objek Web tertentu dalam lingkungan cache. Ketika sebuah cache mengirimkan query ICP ke tetangganya, tetangga mengirim kembali reply ICP yang mengindikasikan "HIT" atau "MISS" [DWC-97].

ICP terutama dipergunakan dalam hirarki cache untuk mencari objek tertentu dalam cache sibling. Hal ini dilakukan ketika cache squid tidak menemukan objek yang direquest, maka proxy ini akan mengirim query ICP ke

siblingnya dan sibling akan merespon dengan jawaban ICP yang mengindikasikan HIT atau MISS.

2.4 PEMROGRAMAN SOCKET

2.4.1 Socket

Sebuah socket didefinisikan sebagai identifikasi unik ke atau dari mana informasi ditransmisikan dalam network [JMW-71].

Pemrograman socket diperlukan dalam menghubungkan dua buah komputer untuk berkomunikasi satu sama lain. Socket merupakan jembatan yang menentukan jenis komunikasi antara kedua komputer.

Fungsi socket, -- seperti set up dan mengakhiri koneksi TCP, mengirim dan menerima paket UDP -- dirancang sebagai transport indipenden, dimana alamat protokol dilewatkan sebagai argumen fungsi. Alamat protokol ini dibawa melalui pointer yang tertutup.

Jenis hubungan ini tergantung pada domain, family dan type socket. Tipe-tipe dari socket antara lain:

1. SOCK_STREAM

Type ini memberikan stream byte yang *connection base* (berbasis koneksi), *duplex* (2 arah), *reliabel*, dan *sekwensial* (berurutan). Memungkinkan mekanisme transmisi data yang melebihi batas.

2. SOCK_DGRAM

Mendukung datagram dengan sifat-sifat *connectionless* (tanpa koneksi), *unreliabel* (tidak ada jaminan sampainya data), dengan panjang data maksimum yang sudah fix/pasti.

3. SOCK_SEQPACKET

Memberikan transmisi *duplex* (dua arah), *connection base* (berbasiskan koneksi) untuk datagram yang memiliki panjang maksimum yang sudah ditentukan. *Sekwensial* dan *reliabel*.

4. SOCK_RAW

Dipergunakan untuk akses protokol jaringan *raw*.

5. SOCK_RDM

Menyediakan lapisan datagram *reliabel* yang tidak menjamin *ordering*.

6. SOCK_PACKET

Tipe ini sudah tidak dipakai lagi. Beberapa tipe socket tidak dapat diterapkan oleh semua famili protokol; misalnya, `SOCK_SEQPACKET` tidak diterapkan pada `AF_INET`.

Protokol menentukan aturan yang dipergunakan oleh socket. Biasanya keberadaan sebuah protokol mendukung satu tipe socket tertentu. Akan tetapi bisa saja beberapa protokol ada dalam satu socket.

Socket dengan type `SOCK_STREAM` adalah *stream* byte yang *full duplex* (berjalan dua arah). Sebuah socket *stream* harus berada dalam keadaan terkoneksi (*connected state*) ketika menerima atau mengirim data.

Protokol komunikasi ini memastikan bahwa tidak ada data yang hilang atau tergandakan. Apabila terdapat sepenggal data yang tidak dapat terkirimkan pada waktu yang telah diberikan, maka koneksi ini dimatikan.

SOCK_DGRAM memberikan kemampuan mengirimkan paket datagram. Data ini diterima dengan fungsi `recvfrom()`.

2.4.2 Bind

Fungsi bind memberi nama ke suatu socket. Proses ini diterapkan oleh fungsi `bind(int s, struct sockaddr myaddr, socklen_t addrLen)`. ketika sebuah socket dibangun dengan fungsi `socket()`, socket yang terbentuk belum memiliki nama yang digunakan untuk koneksi.

Nilai kembalian fungsi ini apabila berhasil adalah 0, sedangkan jika tidak berhasil adalah –1.

2.4.3 Connect

Koneksi adalah rangkaian lapisan transport yang dibuat antara dua program dengan tujuan untuk komunikasi [NWG-99] . Connect memprakarsai koneksi sebuah koneksi pada socket. Hal ini diterapkan dengan fungsi `connect(int sockfd, const struct sockaddr * servaddr, socklen_t addrlen)`

Parameter `sockfd` adalah socket yang telah dibangun., apabila socket bertipe SOCK_DGRAM, maka `servaddr` adalah alamat dimana datagram hendak dikirim dan diterima. Apabila socket bertipe SOCK_STREAM atau SOCK_PACKET.

maka fungsi ini akan mencoba untuk membuat sebuah koneksi ke socket yang lain. Socket lain tersebut dispesifikasikan oleh `servaddr` yang merupakan sebuah alamat dengan panjang `addrlen`.

Pada umumnya, socket berbasiskan koneksi berhasil melakukan koneksi sekali saja; sedangkan socket protokol *connectionless* dapat menggunakan connect berkali-kali untuk berganti pasangan koneksi.

Fungsi ini mengembalikan nilai 0 jika sukses dan –1 jika terjadi error.

2.4.4 Listen

Fungsi `listen(int s, int backlog)` mendengarkan (menunggu) apabila terjadi koneksi pada socket. Untuk menerima sebuah koneksi, sebuah socket pertamakali dibuat terlebih dahulu dengan fungsi `socket()`, dan kesediaan untuk menerima koneksi serta menentukan batas jumlah antrian yang datang adalah fungsi `listen()`, dan kemudian koneksi yang datang diterima dengan fungsi `accept()`. Fungsi ini hanya diterapkan pada type `SOCK_STREAM` atau `SOCK_PACKET`.

2.4.6 recv()

Fungsi `recv()` menerima message dari socket, dan dapat dipergunakan untuk menerima data baik itu dalam keadaan terkoneksi maupun tidak terkoneksi. Apabila tidak terdapat message yang datang, maka rutin ini akan menunggu message datang, kecuali jika socket ini nonblocking.

2.5 Terminology

Alamat IP (IP Address) adalah identifier bagi sebuah komputer atau piranti pada sebuah jaringan TCP/IP. Jaringan yang mempergunakan protokol TCP/IP melewatkan pesan berdasarkan alamat IP tujuan. Format dari alamat IP adalah bilangan 32 bit, ditulis sebagai 4 bilangan yang dipisahkan dengan titik. Masing-masing bilangan berkisar antara 0 hingga 256 [JPM-03].

Bandwidth adalah jumlah data yang dapat ditransmisikan dalam satu satuan waktu tertentu. Biasanya dinyatakan dalam satuan bit per detik (bps = bits per second), atau byte per detik (Bps = bytes per second), atau Hertz. Bandwidth sangat penting bagi peralatan I/O, misalnya sebuah media disk yang cepat dapat dihalangi oleh sebuah bus yang memiliki bandwidth rendah [JPM-03].

Bitmask adalah bilangan 32 bit, baik sebagai sebuah bilangan tak bertanda maupun sebagai satu set bit. Misalnya bilangan 13 dapat direpresentasikan sebagai bitmask 13 (sebagai bilangan desimal), bitmask 0b1101 (sebagai bilangan biner), atau b0 + b2 + b3 (sebagai bit).

Block penggunaan sumber daya oleh suatu poses dengan menutup kemungkinan proses lain mempergunakan sumber daya tersebut. Sumber daya dapat berupa data atau device (piranti keras).

Buffer adalah tempat penyimpana sementara, biasanya di dalam RAM. Kebanyakan bertujuan sebagai tempat dilangsungkannya proses, yang memungkinkan CPU memanipulasi data sebelum mentransfernya ke dalam piranti [JPM-03].

Event adalah aksi atau peristiwa yang dideteksi oleh program. Event dapat berupa aksi dari pemakai seperti klik tombol mouse atau penekanan keyboard, atau kejadian pada sitem, seperti running out of memory.

File Descriptor adalah sebuah Integer yang mengidentifikasikan sebuah file yang dibuka oleh sebuah proses. Bilangan ini didapatkan sebagai hasil dari membuka sebuah file. Operasi yang membaca, menulis dan menutup file harus mempergunakan file descriptor sebagai parameter inputnya [WMB-98].

Fork adalah fungsi yang membuat proses child yang memiliki PID dan PPID berbeda dengan proses parentnya, dan penggunaan resourscenya diset 0, lock file dan sinyal pending tidak diturunkan [LPM-95].

FQDN (Fully Qualified Domain Name) adalah nama domain yang memenuhi kualifikasi penamaan [JPM-03], yaitu terdiri atas nama host dan nama domain, termasuk domain level atas. Misalnya www.squid-cache.org adalah FQDN dimana host adalah host, squid-cache adalah nama domain level kedua, dan org adalah nama domain level atas. Sebuah FQDN selalu dimulai dengan nama host dan dilanjutkan dengan level-level berikutnya hingga mencapai level paling atas.

Fungsi callback adalah fungsi yang diberikan ke fungsi lainnya, untuk dipanggil ketika fungsi tersebut berakhir / selesai. Fungsi ini dipergunakan ketika memulai sebuah aktifitas yang membutuhkan beberapa waktu untuk selesai, dan pengguna menginginkan agar squid melakukan hal lain [RCL-00].

Gateway adalah simpul di dalam suatu jaringan komputer (network) yang menjadi pintu masuk ke dalam jaringan komputer lainnya

Handler adalah fungsi atau metode yang berisikan statemen program yang dieksekusi sebagai respon dari suatu event.

Host adalah sebuah komputer yang terhubung ke sebuah jaringan komputer [WMB-98].

Hot Objects adalah objek yang paling baru direquest, berapapun jumlahnya yang ada dalam memory [DWS-00].

Hyperlink adalah elemen dalam dokumen elektronik yang menghubungkan dengan tempat lainnya pada dikumen yang sama atau pada keseluruhan dokumen yang berbeda. Biasanya dengan melakukan klik pada hyperlink untuk mengikuti tempat yang ditujunya.

ICMP (Internet Control Message Protocol) adalah ekstensi dari Internet Protocol (IP) yang didefinisikan oleh RFC792 [JPM-03], ICMP mendukung paket yang berisikan pesan-pesan error, kontrol, dan informasi. Sebagai contoh, perintah ping menggunakan ICMP untuk menguji koneksi Internet.

Lifetime adalah lamanya waktu yang dimiliki oleh sebuah klien untuk tetap terkoneksi dengan proses cache. Hal ini dipergunakan untuk mencegah agar cache tidak memiliki begitu banyak socket (dan juga file descriptor) terikat dengan state `CLOSE_WAIT` dari klien yang pergi tanpa melakukan shut down. Defaultnya adalah satu hari atau 1440 menit [RNL-01].

LRU (Last Recently Used) adalah aturan yang dipergunakan dalam paging sistem yang sebuah page untuk dipaging apabila telah dipergunakan (dibaca atau ditulis) paling baru daripada halaman lainnya [WMB-95].

MD5 (Message Digest) adalah algoritma untuk memperoleh inti pesan (message digest) dari sembarang jumlah bit input menjadi 128 bit output, dan merupakan metode yang dipergunakan untuk menguji integritas data [RLR-91].

Metadata adalah informasi yang dapat dipahami oleh mesin dalam lingkunagn web [RLS-01].

MIME (Multipurpose Internet Mail Extension) adalah standar untuk email multimedia dan dokumen-dokumen www di Internet [WMB-95]. MIME menyediakan untuk mentransfer data nontextual, seperti grafik, audio dan fax.

Paging adalah teknik yang dipergunakan untuk meningkatkan ruang memori yang dapat dipergunakan dengan memindahkan bagian yang jarang dipergunakan dari RAM ke media penyimpanan kedua, misalnya disk. Unit dari transfer ini dinamakan page.

Space ruang simpan dalam media penyimpan seperti hard disk atau memory.

StoreEntry adalah struktur, tempat dimana setiap obyek disimpan. Struktur ini disimpan dalam sebuah tabelyang bernama tabel hash. Dengan adanya tabel ini, squid dapat mencari obyek yang dicache dengan cepat.

System Call adalah mekanisme yang dipergunakan oleh aplikasi program untuk melakukan request service dari sistem operasi [WMB-95]. System Call

seringkali mempergunakan kode instruksi mesin yang menyebabkan prosesor berubah mode (misalnya dari mode supervisor menjadi mode terproteksi). Hal ini menyebabkan sistem operasi melakukan aksi pembatasan seperti mengakses peralatan hardware atau unit manajemen memori.

Table hash adalah suatu dictionary, dimana kunci dipetakan ke posisi-posisi array oleh sebuah fungsi hash [PEB-03]. Memiliki lebih dari satu peta kunci ke posisi array dinamakan collision, akan tetapi kunci-kunci ini mungkin dibagi menjadi pengalamatan terbuka, di mana semua elemen disimpan dalam sebuah tabel, dan chaining, dimana struktur data tambahan dipergunakan.

Time Out adalah keadaan dimana suatu piranti telah menyediakan waktu tertentu untuk mendapatkan sinyal tertentu, akan tetapi tidak memperolehnya dalam waktu maksimal yang disediakan. Banyak program yang mempergunakan time out sehingga tidak idle selagi input sinyal tidak pernah datang.

URI (Uniform Resource Identifier) adalah sebuah string yang terdiri atas karakter yang pasti untuk mengidentifikasikan sebuah sumber daya fisik [BFI-98]. URI menyediakan makna yang sederhana dan berekstensi untuk mengidentifikasi sebuah sumber daya. Uniform membolehkan tipe identifier yang berbeda dipergunakan dalam konteks yang sama, bahkan ketika mekanisme yang dipergunakan untuk mengakses sumber daya tersebut berbeda. Resource atau sumber daya adalah sesuatu yang

memiliki identitas. Identifier adalah obyek yang dapat beraksi sebagai referensi ke sesuatu yang memiliki identitas. Dalam kasus URI, obyek adalah urutan karakter dengan sintaks yang tertentu.

URL (Uniform Resource Locator) adalah sebuah syntax dan semantik informasi formal bagi lokasi dan akses sumber daya di Internet [TBL-94].

BAB III

SQUID SEBAGAI PROXY SERVER

BAB III

SQUID SEBAGAI PROXY SERVER

3.1 PENDAHULUAN

Squid adalah sebuah aplikasi cache WWW yang dikembangkan oleh *National Laboratory for Applied Network Research* beserta anggota komunitas Web Caching [WSO-00]. Squid diimplementasikan sebagai sebuah proses single, setiap request ditangani oleh proses utama, dengan perkecualian pada FTP. Squid bekerja *non-block*, berdasarkan pada loop `select()` BSD.

Squid bekerja sebagai perantara. Menerima request dari klien (misalnya browser) dan melewatkannya ke server Internet yang sesuai. Squid menyimpan salinan dari data yang dikembalikan dalam sebuah cache di dalam disk. Keuntungan dari penggunaan Squid dirasakan ketika data yang sama direquest beberapa kali. Karena salinan dari data yang direquest terdapat dalam disk, maka akses Internet menjadi cepat, selain itu juga dapat menghemat *bandwidth*.

Squid terdiri atas sebuah program utama server squid, sebuah program lookup DomainName System *dnsserver*, beberapa program opsional untuk menuliskan request dan melakukan otentikasi misalnya `client.c`, dan beberapa tool klien dan manajemen misalnya file konfigurasi squid.

Terdapat beberapa alasan yang melandasi penggunaan squid sebagai bahan dalam pembuatan Tugas Akhir ini. Alasan-alasan tersebut antara lain :

1. Squid adalah perangkat lunak yang dapat diperoleh secara gratis

2. Kode squid memungkinkan untuk dimodifikasi
3. Terdapat banyak orang yang menggunakan squid, dan bahkan memiliki mailing listnya sendiri
4. Karena beberapa alasan banyak pengguna produk proxy komersial yang berpindah ke squid
5. Pengguna squid membayar untuk support / dukungan, bukan untuk produk
6. Kebutuhan hardware standard, fleksibel dalam memenuhi kebutuhan hardware

3.2 KOMPONEN DALAM SQUID

3.2.1 Client Side

Merupakan tempat koneksi klien baru diterima, diparsing dan diproses. Client Side menentukan apakah sebuah request merupakan cache HIT, REFRESH, atau MISS. HIT berarti bahwa halaman web telah tersimpan dalam cache. REFRESH berarti bahwa user melakukan refresh (reload). MISS berarti bahwa request telah tersimpan dalam cache.

3.2.2 Server Side

Rutin-rutin disini bertanggung jawab bagi kegiatan forward cache miss ke server lain bergantung pada jenis protokol. Cache miss dapat diforward baik itu ke origin server-nya maupun ke proxy cache lainnya.

3.2.3 Storage Manager

Storage Manager adalah perekat antara `Client Side` dengan `Server Side`. Setiap objek yang disimpan dicache dialokasikan ke struktur `StoreEntry`. Sementara objek sedang diakses, dia juga memiliki struktur `MemObject`.

Squid dapat mencari objek yang tersimpan dalam cache karena Squid memiliki *tabel hash* dari semua `StoreEntry` yang ada. Kunci dari *tabel hash* ini adalah checksum *MD5* dari objek *URI*. Terdapat pula aturan penyimpanan seperti *LRU* untuk menjaga track objek dan memutuskan objek mana yang harus dihapus apabila diperlukan *space*. Data objek meliputi HTTP Reply penuh – header dan body.

`StoreEntry` memetakan masing-masing objek ke `cache_dir` dan lokasi melalui `sdirn` dan `sfilen`. Untuk menyimpan file ufs, nomor file ini (`sfilen`) dikonversikan ke sebuah nama path dengan sebuah modulo sederhana L1 dan L2. Akan tetapi driver storage lainnya mungkin memetakan `sfilen` dengan cara yang berbeda. Sebuah file cache swap terdiri atas dua bagian: cache *metadata* dan objek data (full HTTP Reply – headers dan body).

Request dari `client side` meregistrasi dengan sebuah `StoreEntry` yang diberitahukan ketika data baru tiba. Beberapa klien dapat menerima data dari single (satu) `StoreEntry`.

3.2.4 Access Control

Fungsi-fungsi di sini bertanggung jawab untuk mengijinkan atau menolak request berdasarkan pada jumlah parameter yang berbeda. Parameter-parameter

ini meliputi alamat IP client, nama host dari sumber daya yang direquest, metode request, dll. Beberapa informasi penting mungkin tidak dapat berfungsi secara langsung, misalnya alamat IP server.

3.2.5 Manajer Cache

Manager cache menyediakan akses terhadap informasi tertentu yang dibutuhkan oleh administrator cache.

3.2.6 Event Queue

Memelihara sebuah link list bagi antrian event yang hendak dilaksanakan di waktu berikutnya.

3.3 PROGRAM-PROGRAM EKSTERNAL

3.3.1 dnserver

Squid menggunakan proses eksternal untuk melakukan pemanggilan terhadap `gethostbyname()`, karena fungsi ini melakukan *block*. Biasanya terdapat sepuluh proses dnsserver.

3.3.2 unlinkd

System Call `unlink` dipakai untuk menghapus nama pada filesystem. Apabila nama tersebut adalah link terakhir ke sebuah file dan tidak ada proses yang membukanya, maka file tersebut dihapus dan *space* yang telah selesai dipergunakannya dapat dipergunakan kembali untuk keperluan lainnya.

Apabila nama tersebut adalah link terakhir ke sebuah file namun terdapat proses yang menggunakannya, file tersebut akan tetap ada hingga *file descriptor* terakhir yang mereferensi kepadanya ditutup.

Apabila nama tersebut direferensikan ke sebuah socket, fifo, atau device, nama akan dihapus namun proses yang membuka objek tetap menggunakannya.

System call `unlink` pada sistem operasi ini tidak dipakai dalam squid karena dapat menyebabkan proses mengalami *blocking*. Oleh karenanya squid mempergunakan `unlinkd` untuk keperluan ini.

3.3.3 redirector

Redirector adalah sebuah program eksternal yang memiliki kemampuan untuk menulis ulang *URL* yang direquest. Redirector dapat dipergunakan oleh administrator untuk mengetahui ke lokasi manakah pengguna Internet melakukan akses.

Setelah melakukan cek terhadap access control, akan tetapi sebelum melakukan cek terhadap cache hit, *URL* yang direquest bisa ditulis di sebuah proses redirector external

3.4 ALUR DARI SEBUAH REQUEST

Alur dari sebuah dapat dijelaskan sebagai berikut :

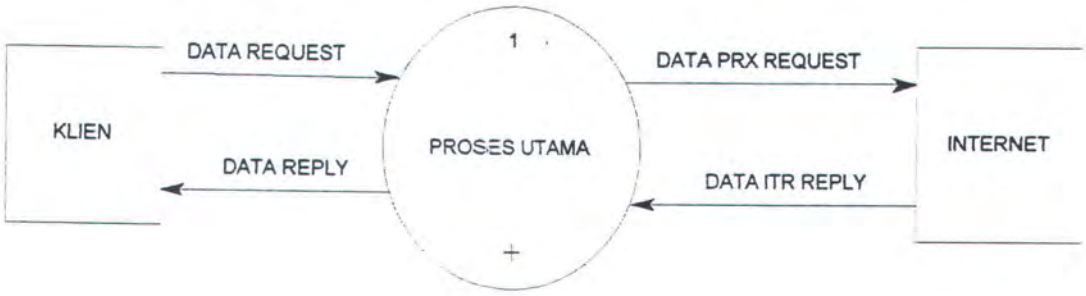
1. sebuah koneksi klien diterima oleh *client_side*, request HTTP diparsing.

2. *Access Control* diperiksa. Client-side membangun sebuah struktur data ACL dan meregistrasikan sebuah *fungsi callback* untuk notifikasi ketika pengecekan access control selesai.
3. Setelah memastikan access control, *client-side* mencari objek yang bersangkutan dalam cache. Apabila sebuah request *hit*, maka *client-side* meregitrasikannya ke *StoreEntry*, Squid memforward request tersebut.
4. Request *client side* memforward stream klien ke *GetMoreData* yang mencari objek bersangkutan di dalam cache. Apabila status yang dilaporkan adalah *cache-hit*, *client-side* meregisterkannya ke dalam registry, sebaliknya squid memforward request tersebut.
5. Proses *request-forwarding* dimulai dengan *ProtoDispatch*. Fungsi ini memulai prosedur pemilihan peer, yang meliputi pengiriman query icp, dan menerima jawaban icp. Prosedur pemilihan peer juga meliputi pengecekan opsi konfigurasi seperti *never_direct* dan *always_direct*.
6. Ketika reply icp (jika ada) telah diproses, kita berhenti di protostart. Fungsi ini memanggil fungsi protokol-spesifik yang sesuai untuk memforward request tersebut. Di sini kita mengasumsikan bahwa ini adalah request HTTP.
7. Modul HTTP mula-mula membuka sebuah koneksi ke origin server atau cache peer. Apabila tidak terdapat soket yang dapat dipergunakan, sebuah koneksi baru diberikan ke modul Network Communication dengan sebuah *fungsi callback*. Rutin `comm.c` mencoba membangun koneksi beberapa kali.
8. Ketika sebuah koneksi TCP telah dibangun, HTTP membangun sebuah *buffer* request dan submit untuk menuliskannya ke dalam socket. Protokol ini

kemudian meregistrasikan sebuah penanganan pembacaan untuk menerima dan memproses reply HTTP.

9. Ketika reply diterima pertama kali, header reply HTTP diparsing dan ditempatkan ke dalam struktur data reply. Bersamaan dengan dibacanya data reply, data ini dibubuhkan ke `StoreEntry`. Setiap kali data dibubuhkan ke `StoreEntry`, `clientSide` diberitahukan tentang data yang baru melalui sebuah *fungsi* `callback`.
10. Ketika `client-side` diberitahukan tentang adanya data baru, `client-side` menyalin data dari `StoreEntry` dan mengajukannya untuk penulisan pada soket klien
11. Ketika data dicantumkan ke `StoreEntry`, dan klien membacanya, data tersebut mungkin diajukan untuk penulisan ke disk.
12. Ketika modul HTTP telah selesai membaca balasan dari upstream server, protokol ini menandai `StoreEntry` sebagai "complete". Soket server ditutup atau diberikan ke pool koneksi untuk penggunaan di lain waktu.
13. Ketika *client side* telah menuliskan semua data objek, dia meng-unregister dirinya dari `StoreEntry`. Pada saat yang sama dia dapat menunggu request lain dari klien, atau menutup koneksi klien.

Berikut adalah diagram alur data dari proses internal squid :



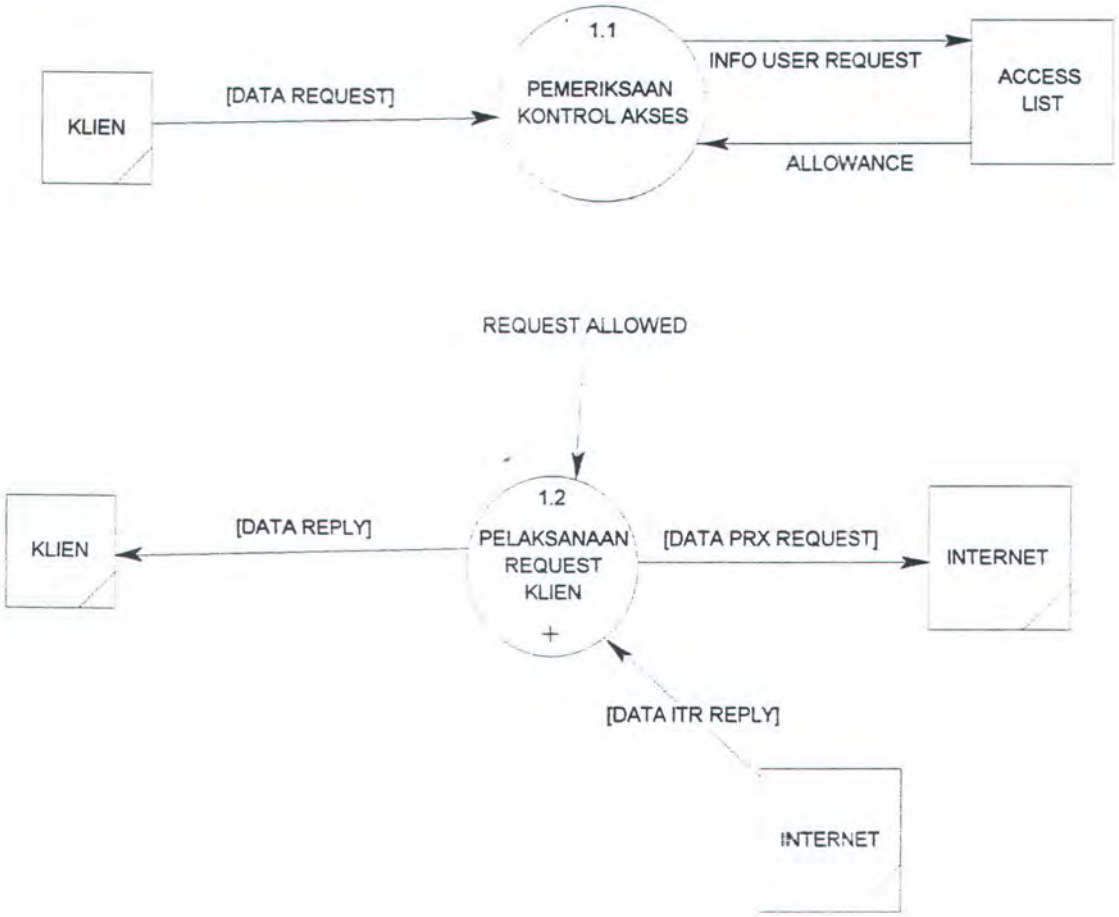
Gambar 3.1 DFD Level 1

Pada gambar 3.1 di atas, proses utama squid memiliki relasi dengan dua buah entitas, yakni klien dan Internet. Kedua entitas ini masing-masing memiliki hubungan yang timbal balik dengan proses squid, yaitu request dan reply.

Entitas klien mewakili pengguna akhir Internet yang mempergunakan server proxy untuk mengakses Internet. Server proxy digambarkan sebagai sebuah proses utama, menerima data request dari klien, dan mengembalikan data reply ke klien apabila telah mendapatkannya.

Hubungan dengan Internet hanya terjadi apabila data yang direquest oleh klien tidak diperoleh di dalam proxy (cache).

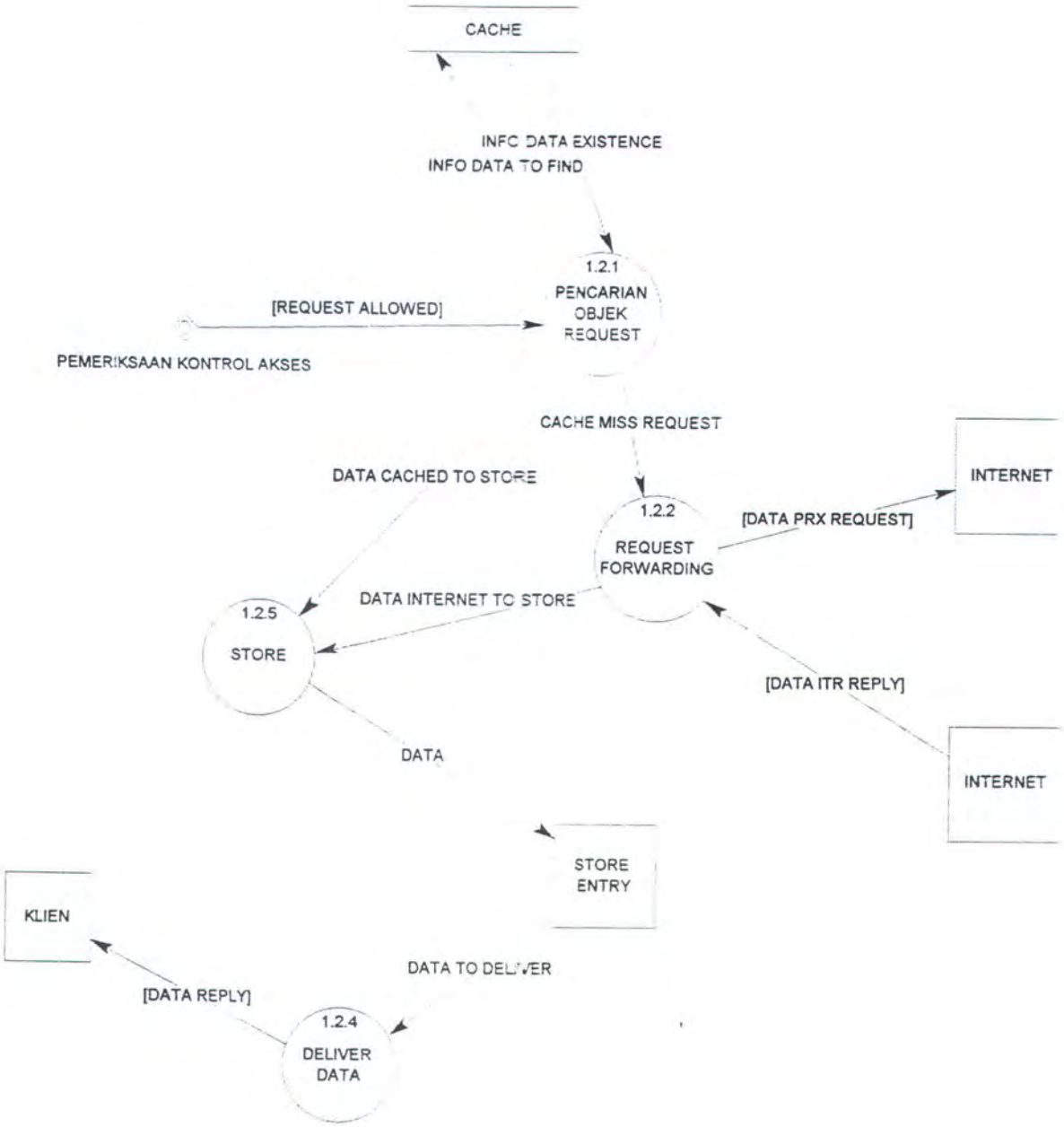
Proses utama pada level satu ini dapat didecompose (dipecah) menjadi level dua yang menjelaskan kejadian proses menjadi lebih detil sebagaimana pada gambar 3.2 berikut :



Gambar 3.2 DFD Level 2

Dalam level dua ini, pemeriksaan kontrol akses dipergunakan untuk memeriksa sejauh mana hak-hak pengguna Internet untuk mengakses objek tertentu dari Internet berdasarkan pada data requestnya. Apabila daftar akses (access list) memperkenankan request dari pengguna, maka request yang diperbolehkan akan di teruskan ke proses selanjutnya.

Pelaksanaan request klien dapat dipecah menjadi beberapa proses dalam level tiga sebagai berikut :



Gambar 3.3 DFD Level 3 Proses Internal Squid

Setelah diperiksa oleh proses kontrol akses, request data diteruskan ke proses pencarian objek request. Proses ini memeriksa cache, apabila terdapat

objek yang diminta oleh klien, maka objek disimpan ke `StoreEntry` melalui proses Store.

Apabila proses pencarian objek request tidak berhasil menemukan objek yang dimaksud, maka pencarian diteruskan ke Internet melalui proses request forwarding. Proses ini meminta kepada sistem luar data-data yang direquest oleh klien namun tidak diperoleh di cache lokal. Apabila usaha dari proses ini berhasil, maka data yang diperoleh pun akan disimpan ke `StoreEntry` melalui proses Store.

Proses Deliver Data masih merupakan bagian dari proses utama Squid, yaitu pada bagian `Client-Side`. Proses ini menyalinkan data dari `StoreEntry` ke socket klien begitu diberitahukan adanya data reply yang datang.

3.5 LOOP UTAMA

squid tidak mempergunakan *fork* dalam menangani request kliennya. Hal ini disebabkan karena penggunaan *fork* akan mengakibatkan terjadinya *block*ing bagi proses-proses yang terlibat. Untuk menangani adanya banyak request dari klien, maka squid menggunakan fungsi `select()` dan `loop()` yang telah disesuaikan untuk squid sendiri, yaitu `comm_select`.

System Call `select()` dan `loop()` bekerja dengan cara menunggu event-event I/O pada sebuah set *file descriptor*. Squid hanya melakukan cek terhadap event-event read dan write. Hal ini dilakukan apabila terdapat handling write dan read pada *file descriptor* yang diberikan. Fungsi-fungsi handling

diregistrasikan dengan fungsi `commSetSelect`. Contohnya adalah sebagai berikut :

```
commSetSelect(fd,COMM_SELECT_READ,clientReadRequest,conn,0);
```

dalam contoh ini, argumen fd adalah koneksi socket TCP ke sebuah klien. Ketika sebuah data dibaca dari socket (fungsi `commSetSelect` membaca data dari socket karena argumen kedua dari fungsi ini adalah `COMM_SELECT_READ`), maka loop `select()` akan mengeksekusi :

```
clientReadRequest(fd, conn);
```

handling I/O direset setiap kali dipanggil. Dengan kata lain sebuah fungsi penanganan harus di registrasi ulang apabila ingin terus membaca atau menulis pada *file descriptor.*

Untuk menggagalkan aksi handling ini, maka argumen NULL dipergunakan sebagai berikut :

```
commSetSelect(fd,COMM_SELECT_READ,NULL,NULL,0);
```

handling I/O ini, dengan pointer data *callback* disimpan dalam struktur data `fde`

```
struct _fde {
        ...
        PF *read_handler;
        void *read_data;
        PF *write_handler;
        void *write_data;
        close_handler *close_handler;
        DEFER *defer_check;
        void *defer_data;
};
```

Gambar 3.4 struktur fde

Read_handler dan write_handler dipanggil ketika *file descriptor* siap melakukan kegiatan baca tulis. Close_handler dipanggil ketika *file descriptor* ditutup, dan merupakan link list dari fungsi-fungsi *callback* yang dipanggil.

Squid senantiasa memeriksa fungsi defer_check untuk mengetahui apakah squid membatalkan event baca dari sebuah *file descriptor* ataukah tidak, maka nilai fungsi defer_check diperiksa. Apabila nilai ini 1, maka squid melewati *file descriptor* sepanjang loop select() tersebut.

Masing-masing handling ini disimpan dalam struktur FD_ENTRY, dengan tipe PF yang merupakan typedef :

```
typedef void (*PF) (int, void *);
```

Comm_select() adalah fungsi yang mempergunakan system call select(). Comm_select() menscan seluruh array fd_table[] untuk mencari fungsi handling. Masing-masing *file descriptor* dengan sebuah handling read akan diset dalam *bitmask* read fd_set.Dengan cara yang sama handling write discan dan bit-bit diset untuk *bitmask* write. Kemudian fungsi select() dipanggil, dan nilai bitmask read dan write yang dikembalikan discan untuk descriptor dengan I/O yang ditunda.

Setelah masing-masing handler dipanggil, comm_select_incoming, dipanggil untuk memproses request HTTP dan ICP baru.

Yang merupakan handling read adalah httpReadReply(), diskHandlerRead(), icpHandleUdp() dan ipcache_dnsHandleRead(). Sedangkan handling write antara lain adalah commHandleWrite(), diskHandleWrite(), dan icpUdpReply(). Fungsi handler diset dengan

Read_handler dan write_handler dipanggil ketika *file descriptor* siap melakukan kegiatan baca tulis. Close_handler dipanggil ketika *file descriptor* ditutup, dan merupakan link list dari fungsi-fungsi *callback* yang dipanggil.

Squid senantiasa memeriksa fungsi defer_check untuk mengetahui apakah squid membatalkan event baca dari sebuah *file descriptor* ataukah tidak, maka nilai fungsi defer_check diperiksa. Apabila nilai ini 1, maka squid melewati *file descriptor* sepanjang loop select() tersebut.

Masing-masing handling ini disimpan dalam struktur FD_ENTRY, dengan tipe PF yang merupakan typedef :

```
typedef void (*PF) (int, void *);
```

Comm_select() adalah fungsi yang mempergunakan system call select(). Comm_select() menscan seluruh array fd_table[] untuk mencari fungsi handling. Masing-masing *file descriptor* dengan sebuah handling read akan diset dalam *bitmask* read fd_set.Dengan cara yang sama handling write discan dan bit-bit diset untuk *bitmask* write. Kemudian fungsi select() dipanggil, dan nilai bitmask read dan write yang dikembalikan discan untuk descriptor dengan I/O yang ditunda.

Setelah masing-masing handler dipanggil, comm_select_incoming, dipanggil untuk memproses request HTTP dan ICP baru.

Yang merupakan handling read adalah httpReadReply(), diskHandlerRead(), icpHandleUdp() dan ipcache_dnsHandleRead(). Sedangkan handling write antara lain adalah commHandleWrite(), diskHandleWrite(), dan icpUdpReply(). Fungsi handler diset dengan

commSetSelect(), kecuali handler close, yang diset dengan comm_add_close_handler().

Handler close secara normal dipanggil dari comm_close(). Handler ini menghapus struktur data yang diasosiasikan dengan *file descriptor*. Demi alasan ini letak comm_close harus di akhir dari alur program untuk mencegah pembebasan memory.

3.6 CLIENT STREAM

pipe adalah koneksi software antara dua program atau command yang temporer [JPM-03]. Secara normal sistem operasi menerima input dari keyboard dan mengirim output ke layar display. Kadangkala diperlukan menggunakan output dari command satu sebagai input pada command lainnya tanpa melewatkan data melalui keyboard atau layar.

Sebuah clientStream adalah *pipe* berpasangan *uni direksional*, maksudnya adalah sebuah saluran komunikasi yang tidak satu arah.

3.7 STORAGE MANAGER

3.7.1 Store I/O calls

3.7.1.1 storeCreate()

```
storeIOState * storeCreate(StoreEntry *e, STIOCB
*file_callback, STIOCB *close_callback, void *
callback_data)
```

Fungsi ini dipanggil untuk menyimpan `StoreEntry` yang diberikan dalam direktory. *callback* adalah sebuah fungsi yang akan dipanggil baik ketika sebuah error terjadi maupun ketika objek ditutup (dengan memanggil `storeClose()`).

3.7.1.2 storeOpen()

```
storeIOState * storeOpen(StoreEntry *e, STFNCB *
file_callback, STIOCB * callback, void *callback_data)
```

Fungsi `storeOpen()` dipanggil untuk membuka `StoreEntry` yang diberikan dari direktori tempatnya berada. *callback* adalah sebuah fungsi yang akan dipanggil baik ketika sebuah error terjadi maupun ketika objek ditutup (dengan memanggil `storeClose()`). Apabila request open berhasil dilakukan, maka tidak terdapat *callback*. Modul yang memanggil hendaknya mengasumsikan bahwa request yang diopen akan berhasil, dan dapat membaca atau menulis seketika. `storeOpen()` mengembalikan `NULL` jika objek yang direquest tidak dapat dibuka. Dalam kasus ini fungsi *callback* tidak akan dipanggil.

3.7.1.3 storeRead()

```
void storeRead(storeIOState *sio, char *buf, size_t
size, off_t offset, STRCB *callback, void
*callback_data)
```

Fungsi ini lebih rumit daripada fungsi-fungsi lainnya karena membutuhkan *fungsi callback*-nya sendiri untuk memberitahukan kepada pemanggil fungsi ketika data yang direquest telah sedang dibaca. `buff` hendaknya adalah sebuah

buffer memory yang valid pada ukuran byte terkecil. `offset` menentukan offset byte dimana read harus dimulai.

3.7.1.4 storeWrite()

```
void storeWrite(storeIOState *sio, char *buf, size_t
size, off_t offset, FREE *free_func)
```

Fungsi ini mengajukan sebuah request untuk menuliskan satu *block* data ke disk store. Pemanggil fungsi bertanggung jawab dalam mengalokasikan `buff`, akan tetapi semenjak tidak terdapat *callback* per-write, memory ini harus dibebaskan oleh implementasi sistem file yang lebih rendah. Oleh karenanya pemanggil fungsi harus menentukan `free_func` untuk dipergunakan mendealokasikan memory.

3.7.1.4 storeUnlink()

```
void storeUnlink(StoreEntry *e)
```

Fungsi ini menghapus objek tercache dari disk store. Tidak terdapat *fungsi callback*, dan objek tidak perlu dibuka terlebih dahulu. Lapisan sistem file akan menghapus objek apabila terdapat pada disk.

3.7.1.5 storeOffset()

```
storeOffset() off_t storeOffset(storeIOState *sio)
```

Fungsi ini mengembalikan nilai `current_ondisk_offset`. Dipergunakan untuk menentukan berapa banyak memory objek yang dapat dibebaskan untuk memberi jalan bagi objek `in_transit` dan ter-cache lainnya.

`storeIOState` → `offset` mengacu pada offset di disk, jika tidak akan terjadi hasil yang tidak terdefinisikan. Untuk read, fungsi ini mengembalikan offset dari data yang sukses dibaca, tidak termasuk read yang sedang antri.

3.8 KONFIGURASI SQUID

Konfigurasi squid memiliki peranan penting dalam menentukan kinerja squid. Merupakan tugas seorang administrator server proxy untuk memahami dan menjaga konfigurasi ini agar sesuai dengan kebutuhan system.

Program squid membaca file konfigurasi */usr/local squid etc/squid.conf.* Beberapa hal yang penting dalam konfigurasi ini agar squid dapat berjalan adalah

3.8.1 http_port

Penggunaan:

```
http_port port
http_port hostname:port
http_port 1.2.3.4:port
```

Dipergunakan untuk alamat soket dimana Squid mendengarkan request HTTP dari klien.

3.8.2 cache_peer

```
cache_peer hostname type http_port icp_port
```

Untuk memberitahukan apabila memilki sebuah terdapat cache lain dalam hirarki. Peer ini dapat berupa sibling atau parent. Untuk memperoleh hak

menggunakan cache peer ini, harus dilakukan dengan ijin dari administrator peer sebelum mencatumkan alamat peer tersebut.

3.8.3 cache_mem

```
cache_mem (bytes)
```

Cache_mem dipergunakan untuk menentukan jumlah memori yang dipergunakan untuk keperluan caching. Hal yang perlu diketahui oleh administrator adalah bahwa squid mempergunakan jumlah memory jauh lebih banuyak daripada jumlah yang tertera daripada di dalam konfigurasi ini. Apabila kebutuhan sistem akan memory adalah N, maka jumlah yang dituliskan dalam file konfigurasi ini adalah N/3.

Parameter ini tidak menentukan banyaknya memory yang dipergunakan untuk proses. Akan tetapi hanya salah satu aspek dari penggunaan memory squid.

3.8.3 cache_dir

```
cache_dir Type Nama_direktori FS_spesifik_data [opsi]
```

Type menentukan jenis sistem penyimpanan yang dipergunakan. Type jenis ufs paling sering dipergunakan, dan merupakan format penyimpanan squid yang lama. Type jenis aufs merupakan type yang baru, dan dipergunakan untuk mengenablekan async I/O pada Linux atau Solaris.

Nama direktori adalah direktori level atas, dimana file-file swap cache disimpan. Apabila seluruh isi disk hendak dijadikan cache, maka direktori ini

dapat menjadi direktori titik mount . direktori ini harus ada dan dapat ditulis oleh proses squid. Satuan dari memori ini adalah Mega Bytes (MB).

```
cache_dir ufs 100 16 256
```

Banyaknya ruang simpan hard disk yang disediakan untuk keperluan cache adalah 100 MB., 16 buah direktori dengan masing-masing 16 sub direktori dibawahnya. UfS adalah tipe dari sistem file yang dipergunakan.

Direktori cache yang telah ditentukan dalam konfigurasi ini dapat terbentuk setelah perintah berikut ini :

```
/usr/local/squid/bin/squid -z
```

untuk memastikan bahwa perintah ini sudah dijalankan dengan baik dapat dilihat pada file `/usr/local/squid/logs/cache.log`

3.8.4 acl, access_list

acl adalah daftar access control. Berguna untuk mencegah orang yang tidak diharapkan mencuri sumber daya yang kita miliki. Sintak umum dari *acl* ini adalah sebagai berikut :

```
acl aclname acltype string
acl aclname acltype file
```

aclname adalah variabel nama *acl* yang hendak kita pergunakan (user defined). Setiap elemen *acl* mengemban nama *acl* yang unik, dengan kata lain tidak boleh terdapat dua buah *acl* dengan nama yang sama.

Elemen *acl* atau `acltype` adalah tipe dari *acl*. Beberapa diantara tipe-tipe ini antara lain adalah seperti pada halaman berikut :

a. src

untuk alamat IP klien atau range alamat IP. Sintak yang dipergunakan adalah :

acl *aclname* src *ipaddress/netmask* (alamat IP klien)

acl *aclname* src *ipaddress1-ipaddress2/netmask*

(range alamat IP)

di mana

aclname adalah nama *acl* yang dipergunakan.

ipaddress adalah nomor ip dari klien, dan *ipaddress1-ipaddress2* adalah range atau rentang ip klien dari klien dengan alamat IP *ipaddress1* hingga klien dengan alamat IP *ipaddress2*.

Netmask adalah nomor netmask dari jaringan yang memiliki server proxy tersebut.

b. dst

dipergunakan untuk alamat IP host *URL* dengan sintak :

acl *aclname* dst *ipaddress/netmask*

di mana

aclname adalah nama *acl* yang dipergunakan.

ipaddress adalah nomor IP dari host *URL* dan *netmask* adalah netmask dari jaringan tersebut.

c. time

dipergunakan untuk pengaturan waktu, sintak yang dipergunakan adalah sebagaimana pada halaman berikut :

```
acl aclname time    [dayabbrev] [h1:m1 - h2:m2]
```

dimana

aclname adalah nama *acl* yang dipergunakan.

dayabbrev adalah singkatan nama hari dalam bahasa Inggris, yaitu:

S - Sunday

M - Monday

T - Tuesday

W - Wednesday

H - Thursday

F - Friday

A - Saturday

Argumen berikutnya adalah rentang waktu dalam jam dan menit yang dipergunkan di mana *h1:m1* harus lebih besar daripada *h2:m2*

d. port

untuk mengatur port yang diperbolehkan, sintaknya adalah sebagai berikut:

```
acl aclname port 80 70 61
acl aclname port 0-1024
```

di mana

aclname adalah nama *acl* yang dipergunakan.

80 70 61 adalah contoh dari port yang hendak ditentukan nama *acl*nya, sedangkan *0-1024* adalah rentang nomor port

e. `proto`

untuk mengatur port sebagaimana pada point d, akan tetapi penggunaannya berdasarkan nama service sebagai berikut :

`acl aclname proto HTTP FTP`

di mana

aclname adalah nama *acl* yang dipergunakan.

HTTP FTP adalah contoh service yang hendak ditentukan nama *acl*nya

f. `method`

untuk menentukan tipe metode request sebagai berikut :

`acl aclname method GET POST`

dimana

aclname adalah nama *acl* yang dipergunakan.

GET POST adalah contoh dari metode yang hendak ditentukan namanya

g. `maxcon`

sintak yang dipergunakan adalah sebagai berikut :

`acl aclname maxcon number`

aclname adalah nama *acl* yang dipergunakan.

banyaknya koneksi yang diperbolehkan dari sebuah alamat IP . nilainya akan true jika pengguna memiliki koneksi terbuka lebih dari *number*.berbeda dengan elemen *acl*, *access_list* adalah aturan yang terdiri atas *allow* (diperbolehkan) atau *deny* (dilarang), terhadap *acl* yang mengikutinya.

Beberapa *access_list* diantaranya adalah :

a. `http-access`

Memberi ijin kepada klien HTTP (browser) untuk mengakses port HTTP. Sintaknya adalah sebagai berikut :

`http_access allow|deny [!]aclname`

di mana

allow|deny adalah pilihan untuk memperbolehkan atau menolak akses http

aclname adalah nama *acl* yang dipergunakan.

b. `icp_access`

memberi ijin kepada cache tetangga untuk melakukan query ke cache kita dengan ICP. Sintaknya adalah sebagai berikut :

`icp_access allow|deny [!]aclname`

dimana

allow|deny adalah adalah pilihan untuk memperbolehkan atau menolak akses http

aclname adalah nama *acl* yang dipergunakan.

c. `miss_access`

memberikan ijin klien tertentu melakukan forward cache miss melalui cache kita. Sintaknya adalah sebagai berikut :

`miss_access allow|deny [!]aclname`

allow|deny adalah adalah pilihan untuk memperbolehkan atau menolak akses http

aclname adalah nama *acl* yang dipergunakan.

d. always_direct

kontrol dimana request selalu diforward ke origin server web. Sintaknya adalah sebagai berikut :

always_direct *allow|deny* *[!]aclname*

allow|deny adalah adalah pilihan untuk memperbolehkan atau menolak akses http

aclname adalah nama *acl* yang dipergunakan

e. never_direct

kontrol dimana request tidak pernah diforward ke origin server web. Sintaknya adalah sebagai berikut :

never_direct *allow|deny* *[!]aclname*

allow|deny adalah adalah pilihan untuk memperbolehkan atau menolak akses http

aclname adalah nama *acl* yang dipergunakan

f. cache_peer_access

kontrol dimana request dapat diforward ke tetangga yang telah ditentukan (peer). Sintaknya adalah sebagai berikut :

cache_peer_access *allow|deny* *[!]aclname*

allow|deny adalah adalah pilihan untuk memperbolehkan atau menolak akses http

aclname adalah nama *acl* yang dipergunakan

aturan daftar akses dibaca berdasarkan urutan penulisan. Penelusuran list berhenti ketika diketemukan kecocokan *rule*. Apabila terdapat lebih dari satu

elemen *acl* dalam satu rule, dipergunakan logika AND. Apabila tidak ada rule yang cocok, naka aksi defaultnya adalah berlawanan dengan rule terakhir dalam list.

Untuk mengisikan dalam "allowed_host" *acl* kita menggunakan alamat network kita (misalnya 10.126.12.0) dan network mask (misalnya 255.255.255.0).

Contoh dari konfigurasi daftar akses adalah seperti pada gambar 3.5:

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl all src 0.0.0.0/0.0.0.0
acl allowed_hosts src 10.126.12.0/255.255.255.0
http_access deny manager all
http_access allow allowed_hosts
http_access deny all
icp_access allow allowed_hosts
icp_access deny all
```

Gambar 3.5 daftar akses

3.8.5 cache_mgr

```
cache_mgr webmaster
```

Adalah tempat alamat email dari manager cache, untuk tujuan pengalamatan apabila terjadi error.

3.8.6 cache_effective_user

```
cache_effective_user
cache_effective_group
```

Menentukan siapakah yang merupakan pengguna program ini. Apabila squid harus dimulai oleh root, maka perlu dipilih seorang user dan group untuk menjalankan root. Sebaiknya tidak digunakan root untuk alasan sekuriti.

3.8.7 visible_hostname

Yaitu nama host yang digunakan untuk cache, apabila menghendaki nama host khusus yang ditampilkan jika terjadi error.

3.9 FILE LOG SQUID

Squid memiliki file log yang terdapat pada direktori `/usr/local/squid/logs`. File-file log ini berguna untuk mengetahui catatan kegiatan squid dan merupakan umpan balik bagi administrator cache sehingga dapat mengetahui apabila terdapat kesalahan pada program.

File – file log squid yang dipergunakan dalam tugas akhir ini antara lain adalah file access.log dan cache.log.

File access.log memiliki 9 buah field yang dibatasi oleh karakter spasi. Format dari file ini adalah sebagai berikut :

```
Timestamp elapsed src-address type/code/hirarchy size method
URL
```

Field pertama dari file ini adalah waktu Unix dalam satuan mili detik yang dimulai dari tanggal 1 Januari 1970 pukul 00.00 GMT

Field kedua merupakan waktu yang dibutuhkan untuk melakukan download terhadap suatu halaman web

Field ketiga adalah alamat IP dari klien yang melakukan request

Field keempat adalah tipe, yaitu bagaimana request klien ditangani oleh cache

Field kelima adalah code, yaitu kode reply http

Field keenam adalah hierarchy, yaitu kode dari file log hirarki

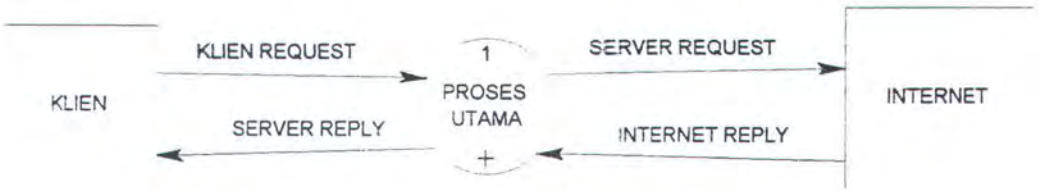
Field ketujuh, size adalah ukuran data yang dituliskan ke klien untuk request TCP. Sedangkan untuk request UDP merupakan ukuran dari request (dalam byte)

Method dalam field kedelapan adalah method yang dipergunakan untuk request (GET, POST, dll)

URL dalam field kesembilan adalah nama *URL* dari request.

3.10 DIAGRAM ALUR DATA

Diagram alur data dari server proxy squid adalah sebagai berikut :



Gambar 3.6 Diagram Alur Data Server Proxy Level 1

Diagram Alur Data level pertama terdiri atas sebuah proses yang bernama Proses Utama dan dua buah entitas luar yang masing-masing bernama Klien dan Internet.

Proses Utama adalah proses utama server proxy yang menghubungkan antara klien web dengan server web (Internet). Proses utama ini menyaring dan meneruskan data terpilih yang masuk dari klien ke Internet.

Klien adalah klien web, yaitu host yang memiliki interface langsung dengan pengguna Internet. Klien melakukan request dan tidak peduli dengan proses data pada server yang ditujunya.

Internet mewakili server web yang sebenarnya, sumber dari data yang direquest oleh klien. Internet diasumsikan dalam kondisi tidak ideal, yaitu sibuk, dan memiliki jalur yang panjang untuk mencapainya.

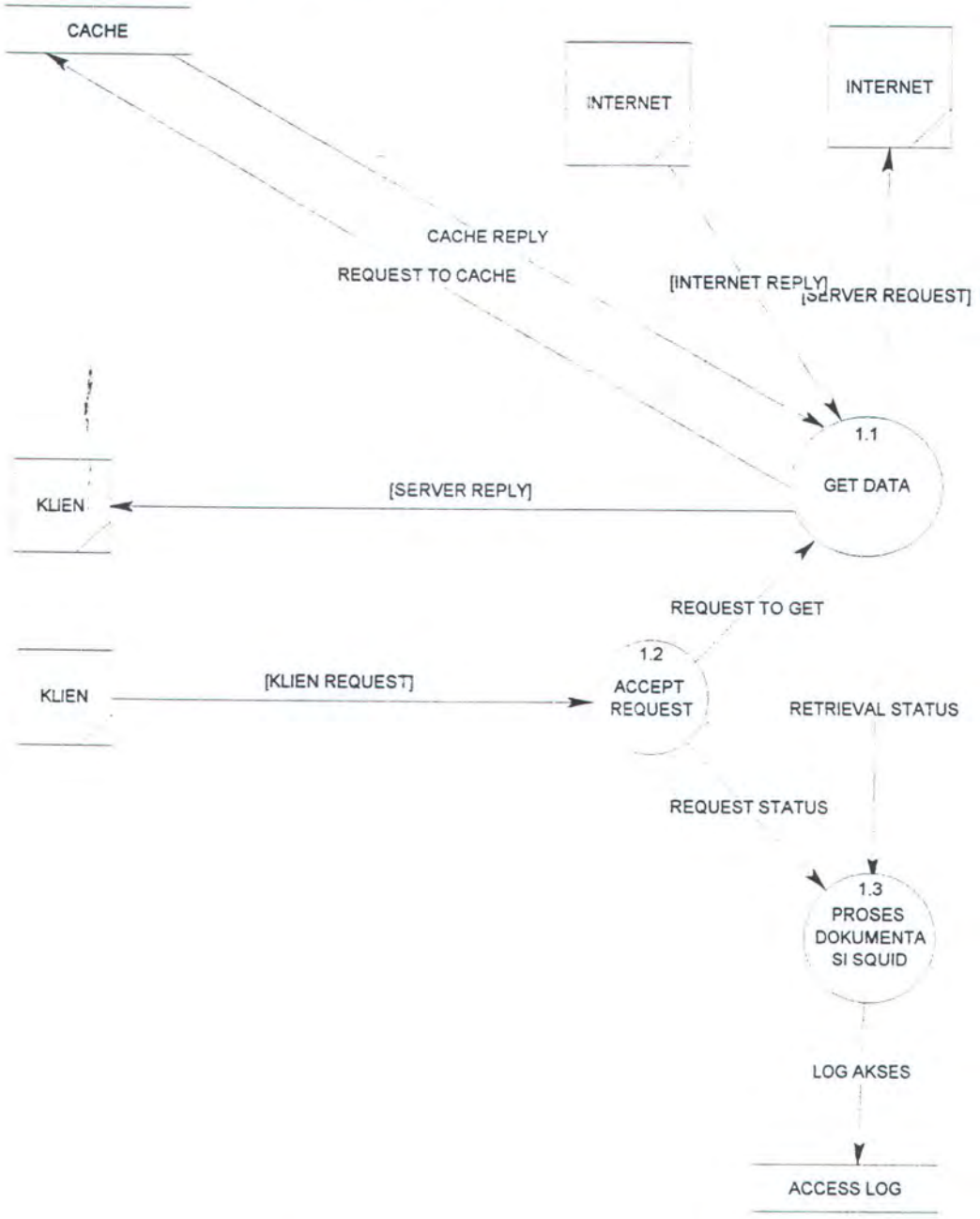
Masing-masing data yang berlalu lintas diantara entitas dan proses di atas adalah data-data yang direquest dan direply sesuai dengan asal entitas atau prosesnya.

Diagram Alur Data level 2 merupakan detil dari proses pada Diagram Alur Data level pertama di atas. Diagram tersebut digambarkan pada gambar 3.7 pada halaman berikut.

Proses ACCEPT REQUEST dan GET DATA dalam program diwakili oleh `client_side`, yaitu bagian dari program utama squid yang berinteraksi dengan sistem luar dengan menerima request dari klien dan mengirimkan data request ke server.

GET DATA meneruskan request yang telah diterima oleh ACCEPT REQUEST baik ke Internet maupun ke cache. Menerima reply dari Internet dan cache, dan meneruskannya ke klien.

Kedua proses ini (ACCEPT_REQUEST dan GET DATA) mengirim data catatan kegiatan proses (log) ke sebuah file yang bernama `access.log`, yang dalam diagram diwakili oleh ACCESS_LOG. Dengan demikian status berhasil atau tidak suatu data diperoleh dapat diketahui.



Gambar 3.7 Diagram Alur Data Server Proxy

CACHE adalah cache server lokal yang menyimpan *metadata* yang pernah direquest oleh klien.

Internet adalah entitas diluar sistem tempat dimana server proxy meminta data apabila dia sendiri tidak dapat memenuhinya secara lokal.

Access.Log adalah sebuah file yang menampung catatan-catatan akses server proxy ini. Catatan ini meliputi antara lain waktu akses, lama load data, nama komputer klien, status akses, metode akses, dan tujuan akses. File ini dibuat oleh proses dokumentasi proses squid

3.11 KEBUTUHAN PERANGKAT KERAS

3.11.1 Memory

Squid sangat membutuhkan memory dalam jumlah besar, hal ini disebabkan karena squid menyimpan *metadata* dari masing-masing objek dalam cache ke dalam memory. *Metadata* ini adalah struktur `StoreEntry`. Terdapat 72 atau 104 byte *metadata* yang tersimpan dalam memory bagi masing-masing objek dalam cache. Sebuah cache dengan 10.000.000 buah objek, membutuhkan 72 MB memori dalam perhitungan, akan tetapi pada prakteknya membutuhkan lebih dari jumlah tersebut.

Squid juga mempergunakan memory untuk menyimpan objek yang transit (squid 1.1), keseluruhan objek disimpan dalam memory hingga transfer lengkap terpenuhi. Selain hal tersebut squid juga menggunakan memory untuk hal-hal seperti pada halaman berikut ini :

- *Buffer* disk untuk baca/tulis, yaitu lokasi di dalam memori yang dipergunakan untuk penampungan sementara objek yang hendak dituliskan atau dibaca
- *Buffer* I/O Network, yaitu lokasi di dalam memori yang dipergunakan untuk penampungan sementara objek yang hendak dilewatkan dalam network
- IP Cache Content, yaitu komponen built-in squid yang melakukan fungsi translasi dari hostname ke nomor IP dan memelihara struktur data yang diperlukan.
- FQDN Cache Content
- Netdb ICMP Measurement data base
- Informasi State per Request, termasuk full request dan reply header
- Kumpulan macam-macam statistik
- "Hot Object", yang disimpan dalam memory, yaitu objek request yang paling baru diterima

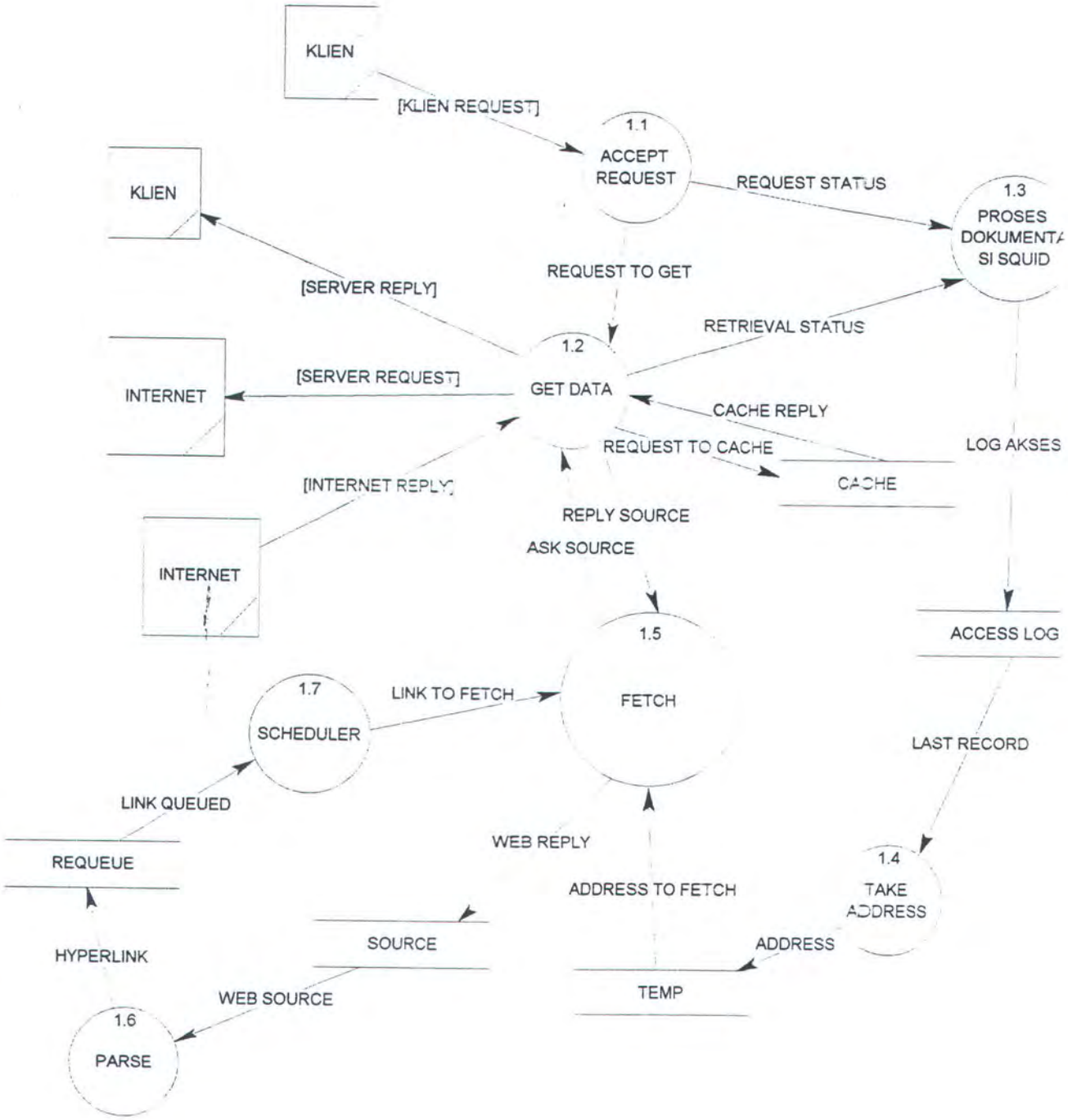
penggunaan memory ini dapat dimonitor dengan perintah `ps -u`, atau `top`.

3.11.2 Prosesor

Squid tidak memerlukan prosesor dengan spesifikasi tinggi, akan tetapi kecepatan yang direkomendasikan adalah 300 MHz. Penggunaan prosesor yang begitu besar adalah pada saat *startup*, selanjutnya penggunaan ini akan menurun pada menit-menit berikutnya.

3.11.3 Hard disk

Kebutuhan akan hard disk berbanding lurus dengan banyaknya data yang akan disimpan. Ketika hendak memilih hard disk untuk keperluan caching, hendaknya mempertimbangkan faktor-faktor *seek time* (waktu pencarian) dan kapasitas penyimpanan.



Gambar 4.2 DFD level 1 Proses Utama Server Proxy Content Cache

Proses CHECK REQUEST pada diagram di gambar 4.2 adalah sama dengan proses yang terdapat pada PROSES UTAMA pada gambar 4.1.

BAB IV

PERANCANGAN DAN IMPLEMENTASI

BAB IV

PERANCANGAN DAN IMPLEMENTASI

Tahap perancangan memberikan kerangka pada tahap implementasi sehingga dapat mengurangi kerumitan dan kesalahan di saat menerapkan pembuatan program. Pada tahap ini diperlukan kejelian dan kemampuan analisa agar pada tahap implementasi tidak terjadi kesalahan fatal. Hal yang perlu dipertimbangkan dalam merancang sistem adalah kebutuhan dari pengguna dan kebutuhan dari sistem itu sendiri.

Pada tahap implementasi, penulisan kode, kompilasi, dan instalasinya dilakukan. Meskipun tidak selalu berhasil, namun pada tahap ini resiko kekeliruan fatal relatif lebih kecil daripada tahap perancangan.

4.1. PERANCANGAN

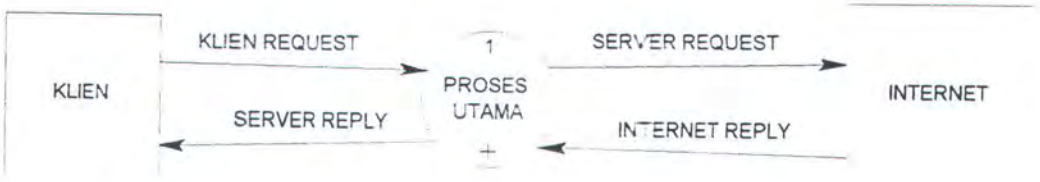
Kebutuhan dari sistem yang telah ada untuk memenuhi keperluan content cache proxy ini adalah program yang mampu membaca halaman Web yang sedang dibuka oleh user, mencari string yang merupakan hyperlink, selanjutnya melakukan method GET untuk memperoleh content dari alamat tersebut. Content message ini disimpan oleh program utama squid di dalam cache.

Program yang dibuat merupakan program eksternal dan senantiasa mendeteksi request dari klien Web. Program ini bernama `showReq`. Setiap kali klien Web melakukan request, `showReq` menyimpan string request ini ke dalam file yang bernama `ReqQueue`. Apabila terdapat request yang sama pada waktu

yang berbeda dari user, maka `showReq` tidak akan menyimpannya kembali ke dalam file `ReqQueue`, karena program sebelumnya telah melakukan pemeriksaan terhadap masing-masing isi file tersebut untuk dicocokkan dengan request klien.

Sebuah program untuk melakukan proses GET ke origin server/sibling/parent bernama fetch membaca satu persatu baris alamat yang disimpan dalam file `ReqQueue`. Setiap kali fetch telah melakukan request terhadap string di dalam file `ReqQueue`, maka string yang bersangkutan akan dipindahkan ke file lain yang bernama `ReqProcessed`. String-string ini disimpan dengan batas maksimal usia yang telah ditentukan. Maksudnya, apabila usia suatu string telah mencapai batas maksimal usianya, maka terdapat program yang menghapusnya. Yaitu program `cleanExpired`.

Fetching data dari origin server memerlukan program klien Web yang mampu melakukan request seperti apa yang telah dilakukan oleh user. Untuk itu dibuat sebuah program internal squid yang bernama `myklien.c` untuk keperluan ini. Kode program ini dibuat dan selanjutnya dikompilasi bersama-sama dengan program utama squid agar dapat menjadi sebuah file eksekusi.



Gambar 4.1 DFD Level 1 bagi request data

4.2 IMPLEMENTASI

Implementasi dari perancangan yang telah dilakukan adalah meliputi persiapan bahan berupa source code hingga pembuatan kode agar dapat bekerja bersama-sama dengan program utama Squid.

Mula-mula harus diperoleh source code Squid proxy server, agar dapat dipelajari dan dimodifikasi kode sumbernya. Dalam tugas akhir ini, source code Squid yang dipergunakan adalah dari Squid2.4 STABLE 6, yang diperoleh dari alamat Internet www.squid-cache.org dalam bentuk tar.gz.

File dengan ekstensi file tar.gz harus diekstrak agar dapat dikompilasi oleh kompiler. Ekstraksi file ini dilakukan dengan perintah gunzip terhadap file squid-2.4STABLE6.tar.gz, sehingga menghasilkan file squid-2.4STABLE6.tar. selanjutnya ekstraksi terhadap tar dilakukan dengan perintah tar –xf squid-2.4STABLE6.tar yang akan menghasilkan direktori squid-2.4STABLE6. di dalam direktori inilah source code dari Squid terdapat dan siap untuk dimodifikasi atau dikompilasi.

File-file yang harus ada dalam proses kompilasi squid ini antara lain adalah : source code, yaitu file-file dengan ekstensi .c dan file-file library dengan ekstensi .h; file configure, yang berisikan script bash aturan konfigurasi squid; dan file makefile, yang berisi aturan untuk mengkompilasi squid serta instalasinya. Agar program tambahan dapat turut serta dikompilasi, maka kode program tersebut disispkan dalam file makefile ini sesuai dengan aturan penulisannya.

Penerapan penambahan kode program ini digambarkan pada gambar berikut :

```
1.  # Gotta love the DOS legacy
2.  #
3.  SQUID_EXE                = squid$(exec_suffix)
4.  CLIENT_EXE               = client$(exec_suffix)
5.  OPT_DNSSERVER_EXE        = dnsserver$(exec_suffix)
6.  DNSSERVER_EXE            =
7.  OPT_UNLINKD_EXE          = unlinkd$(exec_suffix)
8.  UNLINKD_EXE              = $(OPT_UNLINKD_EXE)
9.  OPT_PINGER_EXE           = pinger$(exec_suffix)
10. PINGER_EXE               =
11. CACHEMGR_EXE             = cachemgr$(cgi_suffix)
12. DISKD_EXE                = diskd$(exec_suffix)
13. MYKLIEN_EXE                   = myklien$(exec_suffix)
```

Gambar 4.3 Menentukan nama file eksekusi pada Makefile

Pada gambar 4.3 di atas, nama program myklien disisipkan pada baris terakhir, agar kompiler membuat file eksekusinya. File-file eksekusi ini secara default akan diletakkan pada direktori /usr/local/squid/bin/ sebagaimana file configure telah menentukannya.

```
1.   $(CLIENT_EXE): client.o
2.      $(CC) -o $@ $(LDFLAGS) client.o $(CLIENT_LIBS)
3.
4.   $(MYKLIEN_EXE): myklien.o
5.      $(CC) -o $@ $(LDFLAGS) myklien.o $(CLIENT_LIBS)
6.
7.   $(DNSSERVER_EXE): dnsserver.o
8.      $(CC) -o $@ $(LDFLAGS) dnsserver.o $(DNSSERVER_LIBS)
9.
10. $(CACHEMGR_EXE): cachemgr.o
11.     $(CC) -o $@ $(LDFLAGS) cachemgr.o $(CLIENT_LIBS)
12.
13. $(PINGER_EXE): pinger.o debug.o globals.o
14.     $(CC) -o $@ $(LDFLAGS) pinger.o debug.o globals.o $(PINGER_LIBS)
```

Gambar 4.4 script untuk pembuatan file objek

Pada gambar 4.4 di atas, file objek (file dengan ekstensi .o) ditentukan untuk dibuat pada saat kompilasi. File objek myklien.o terletak pada baris keempat. Untuk membuat file myklien.o ini maka kompiler mengeksekusi baris kelima. LDFLAGS menentukan opsi debug ketika proses kompilasi dijalankan. Opsi ini memproduksi informasi debug, dalam format sistem operasi yang dipergunakan. Sedangkan CLIENT_LIBS adalah library yang diperlukan untuk pembuatan file objek ini.

```
1.  PROGS            = $(SQUID_EXE) $(CLIENT_EXE) $(MYKLIEN_EXE)
2.  UTILS            = $(DNSSERVER_EXE) $(UNLINKD_EXE)
3.  SUID_UTILS       = $(PINGER_EXE)
4.  CGIPROGS         = $(CACHEMGR_EXE)
5.  OBJS             = \
6.                   access_log.o \
7.                   acl.o \
8.                   asn.o \
```

Gambar 4.5 file yang dibuat

Pada gambar 4.5, masing-masing jenis file yang dibuat dikelompokkan dalam kategorinya masing-masing. myklien terletak pada baris pertama dengan kategori program.

Kompilasi squid dilakukan setelah melakukan konfigurasi terhadap file configure. Konfigurasi ini dimaksudkan untuk mengaktifkan atau menonaktifkan fitur-fitur yang ada dalam squid.

Beberapa fitur squid dapat diaktifkan setelah dikompilasi, sehingga harus dilakukan kompilasi ulang pada tahap berikutnya. Suatu fitur memiliki default disable memiliki dua alasan, antara lain adalah sebagaimana pada halaman berikut:

Kompatibilitas Sistem Operasi, meskipun squid ditulis / diprogram secara umum, fungsi-fungsi tertentu tidak dapat bekerja pada sistem operasi tertentu. Ketika banyak sistem operasi yang tidak dapat menggunakan fitur tertentu, pilihan fitur tersebut terdapat pada saat dikompilasi.

Efisiensi, suatu fitur yang menyebabkan kinerja lambat pada kondisi tertentu, perlu didisable.

Salah satu penggunaan konfigurasi yang lain adalah menentukan di direktori mana squid ini hendak diinstall dan opsi-opsi tambahan lainnya, seperti file log yang hendak diinstall. Perintah konfigurasi adalah sebagai berikut :

```
./configure -prefix=/usr/local/squid/
```

Artinya program-program squid akan diletakkan pada direktori /usr/local/squid/ .

Untuk melakukan konfigurasi terhadap lebih dari sebuah opsi, maka masing-masing opsi ditambahkan di akhir baris perintah, misalnya

```
./configure -prefix=/usr/local/squid/ --/enable-async-io
```

Setelah melakukan konfigurasi, selanjutnya dilakukan perintah kompilasi :

```
Make all
```

Kompiler melakukan proses kompilasi terhadap file-file source code squid. Apabila terdapat kesalahan pada source code, maka akan ditampilkan pesan error, dan proses kompilasi akan berhenti, namun apabila proses kompilasi ini berakhir, maka dilanjutkan dengan perintah instalasi :

```
Make install
```

Kompiler melakukan proses instalasi pada direktori yang telah ditentukan dalam proses konfigurasi.

Program-program eksternal dapat diletakkan pada direktori yang disukai oleh user, dalam Tugas Akhir ini, program-program tersebut diletakkan di direktori `/usr/local/squid/ta/`.

Program-program eksternal ini dikompilasi secara terpisah dari file utama squid. Pseudocode dari program-program ini antara lain adalah sebagai berikut :

```
//showReq
// Pseudo code program untuk menampilkan request

1. for (;;){
2.    url=fopen("/usr/local/squid/logs/access.log","r");
3.    fl=fopen("/usr/local/squid/ta/Req","a");
4.    flenn=flen(url);
5.    if (bfr<flenn){
6.      fprintf(fl,noslash(takeaddr(webaddr),webaddrmain));
7.    }
8.    fclose(url);
9.    fclose(fl);
10.   bfr=flenn;
11. }
```

Gambar 4.6 showReq

Program `showReq` di atas memeriksa setiap kali terdapat request dari klien terhadap server proxy, dengan cara memeriksa perubahan pada file lognya. Alamat web yang direquest oleh klien ini disimpan oleh `showReq` dalam sebuah file yang bernama `Req`. Hal ini tentu saja dengan terlebih dulu memeriksa pada file tersebut apakah nama alamat yang hendak disimpan telah ada ataukah belum dalam daftar.

myklien.c

```
1.  /* Build the HTTP request */
2.  for (i = 0; loops == 0 || i < loops; i++) {
3.   /* Connect to the server */
4.   if ((conn=socket(PF_INET, SOCK_STREAM, 0)) < 0) {
5.         exit(1);
6.    }
7.   if (localhost && client_comm_bind(conn,
8.   localhost) < 0) {
9.         perror("client: bind");
10.             exit(1);
11.        }
12.  if (client_comm_connect(conn, hostname, port,
13.ping ?
14.  &tv1 : NULL  < 0) {
15.        if (errno == 0) {
16.fprintf(stderr, "client: ERROR: Cannot connect
17.  to %s:%d: Host
18.   unknown.\n", hostname, port);
19.         } else {
20.            char tbuf[BUFSIZ];
21.            snprintf(tbuf, BUFSIZ, "client: ERROR:
22.   Cannot connect to %s:%d",
23.                     hostname, port);
24.               perror(tbuf);
25.            }
26.            exit(1);
27.        }
28.        /* Send the HTTP request */
29.        bytesWritten = mywrite(conn, msg,
30.strlen(msg));
31.        if (bytesWritten < 0) {
32.            perror("client: ERROR: write");
33.            exit(1);
34.        } else if (bytesWritten != strlen(msg)) {
35.            fprintf(stderr, "client: ERROR: Cannot
36. send request?: %s\n", msg);
37.            exit(1);
38.
38.        }
39.        /* Read the data */
40.   41.webs=fopen("/usr/local/squid/ta/source","w");
41.
```

Gambar 4.7 Pseudocode myclien

```
42.         while ((len = myread(conn, buf,
43.sizeof(buf))) > 0) {
44.             fsize += len;
45.             if (to_stdout)
46.         fwrite(buf, len, 1, webs);
47.         }
48.         (void) close(conn);      /* done
with 49.socket */*/
50.         if (interrupted)
51.             break;
52.     exit(0);
53.     /*NOTREACHED */
54.     return 0;
60.}
```

Gambar 4.8 Pseudocode myclien (lanjutan)

myklien melakukan request web page terhadap server web untuk kemudian disimpan dalam sebuah file yang bernama source, sebagaimana terdapat pada baris ke 34.

4.3 MENJALANKAN PROGRAM

Menjalankan program content cache ini dilakukan setelah program utama squid aktif. File – file yang dibutuhkan untuk keperluan program ini diletakkan pada direktori /usr/local/squid/ta. File – file ini antara lain adalah : main, rf, parseSource, takeAddr, takeLatest.

Untuk menjalankan program ini, file yang harus dieksekusi adalah file main oleh user root sebagai berikut :

```
/usr/local/squid/ta/main
```

Program ini menghasilkan file teks yang bernama `ReqQueue`, yaitu file yang berisikan string-string *hyperlink* dari halaman web yang hendak dieksekusi oleh program content cache.

BAB V

UJI COBA

BAB V

UJI COBA

Uji coba dilakukan terhadap 15 buah situs di Internet., direpresentasikan dalam bentuk tabel empat kolom yang mewakili nomor, alamat Internet, Waktu yang dipergunakan untuk loading data secara sempurna sebelum mempergunakan program, dan waktu yang dipergunakan untuk loading sempurna setelah mempergunakan program. Hasil uji coba adalah sebagai berikut

Terdapat tiga kelompok data yang disimpan dalam tabel yang berbeda. Masing-masing tabel ini memiliki perbedaan level. Level menunjukkan kedalaman akses yang terhadap suatu halaman Web.

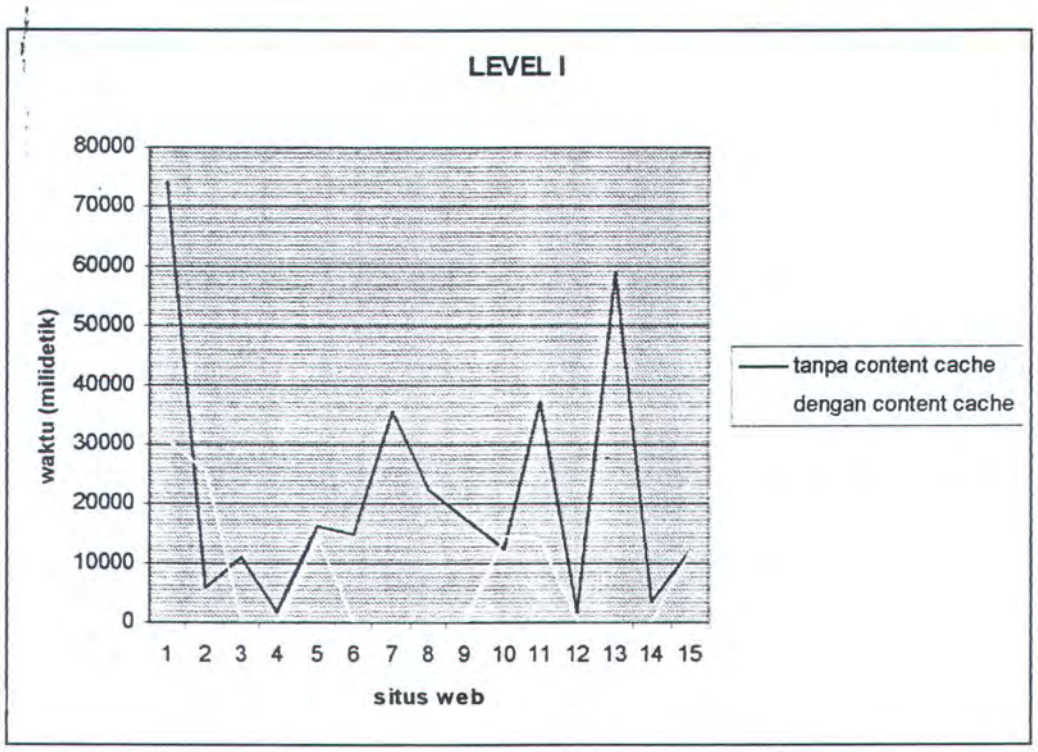
Tabel 5.1 Level 1

No	Alamat	Tanpa Content Cache (dalam mili detik)	Dengan Content Cache (dalam mili detik)
1	www.whitehouse.gov	74197	30592
2	www.cinema.com	5823	25591
3	www.bapepam.go.id	11103	1
4	www.cpan.com	1753	0
5	www.depdagri.go.id	16121	14323
6	www.cia.gov	14888	0
7	www.danamon.co.id	35366	3
8	www.deplu.go.id	22532	0
9	www.assunnah.or.id	17473	0
10	www.bes.com	12322	14597
11	www.alsofwah.or.id	37271	14244
12	www.at-taqwa.com	1806	3
13	www.reuters.com	58815	0
14	www.islamonline.com	3348	0
15	www.liputan6.com	11932	11932

Yang dimaksud dengan level 1 adalah halaman pertama yang muncul ketika pemakai Internet telah mengetikkan alamat Internet di kotak alamat.

Tabel ini terdiri atas empat buah kolom. Kolom pertama adalah nomor dari situs web yang diuji. Kolom kedua adalah nama alamat dari situs web yang diuji. Kolom ketiga adalah waktu yang dipergunakan untuk mendownload seluruh isi dari halaman web dalam satuan mili detik sebelum mempergunakan program content cache. Kolom terakhir adalah waktu yang diperlukan untuk mendownload seluruh isi dari halaman web setelah menggunakan program content cache.

Grafik yang dihasilkan dari tabel di atas adalah sebagai berikut :



Gambar 5.1 grafik pada uji coba pertama

Berdasarkan grafik di atas, diketahui bahwa penggunaan cache memiliki perbedaan jumlah waktu total yang dipergunakan untuk mendownload halaman

web, dibandingkan tidak menggunakan cache. Perbedaan waktu tersebut dapat dihitung dengan perhitungan sebagai berikut :

$$\Lambda = \frac{\sum_{1}^{n}(t2 - t1)}{n} \quad \text{......................................(1)}$$

Dimana :

^ adalah jumlah rata-rata selisih waktu yang dibutuhkan untuk secara penuh mendownload satu halaman web

t2 adalah waktu yang diperlukan untuk mendownload halaman web t sebelum menggunakan content cache

t1 adalah waktu yang diperlukan untuk mendownload halaman web t sesudah menggunakan content cache

n adalah jumlah percobaan yang dilakukan

hasil dari perhitungan menggunakan rumus di atas adalah sebagai berikut:

Σ(t2-t1) = (74197 – 30592) + (5823 – 25591) + (11103 – 1) + (1753 – 0) +

(16121 – 14323) + (14888 – 0) + (35366 – 3) + (22352 – 0) +

(17473 – 0) + (12322 – 14597) + (37271 – 14244) + (1806 – 3) +

(58815 – 0) + (3348 – 0) + (11932 – 11932)

= 213464

n = 15

maka, berdasarkan persamaan (1) :

$$\Lambda = \frac{213464}{15}$$

= 14230.93 mili detik

Tabel 2 berisikan kelompok percobaan pada level 2, dimana halaman yang muncul setelah pengguna melakukan klik terhadap hyperlink pertama pada halaman Web di level 1.

Tabel 5.2 Level 2

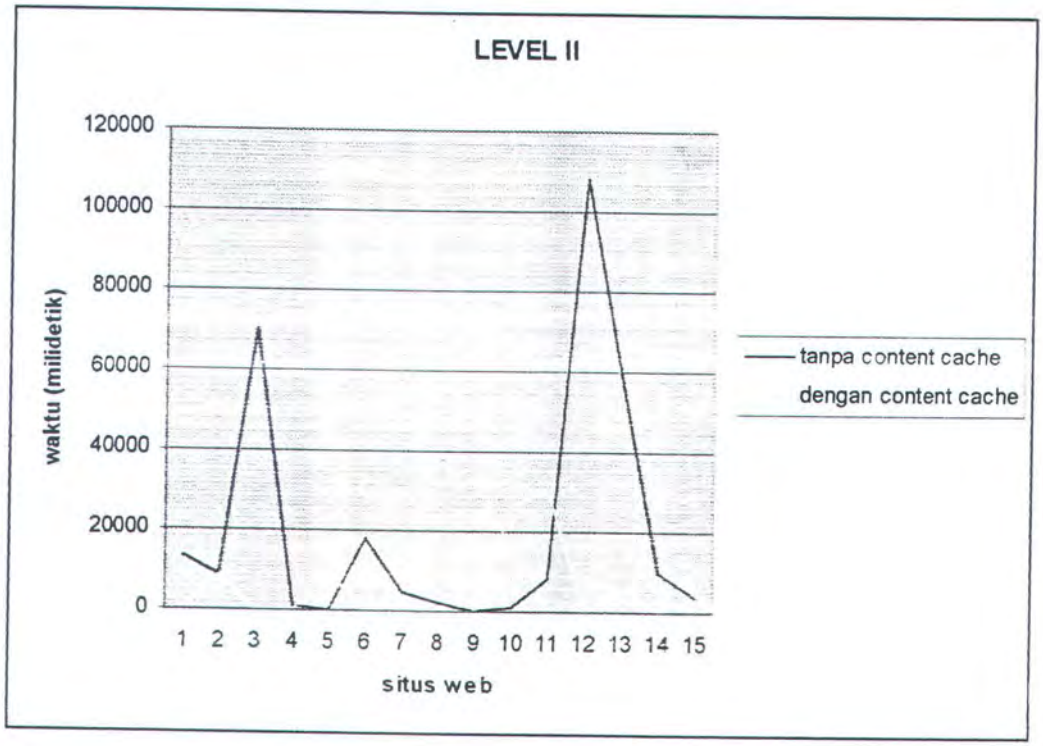
No	Alamat	Tanpa Content Cache (dalam mili detik)	Dengan Content Cache (dalam mili detik)
1	www.whitehouse.gov	13564	1241
2	www.cinema.com	8865	8865
3	www.bapepam.go.id	70033	2
4	www.cpan.com	1051	0
5	www.depdagri.go.id	112	9127
6	www.cia.gov	17906	0
7	www.danamon.co.id	4516	4
8	www.deplu.go.id	2222	0
9	www.assunnah.or.id	249	0
10	www.bes.com	890	26911
11	www.alsofwah.or.id	8457	30442
12	www.at-taqwa.com	108319	12
13	www.reuters.com	58090	5465
14	www.islamonline.com	9497	0
15	www.liputan6.com	3359	2893

Nilai selisih waktu pada uji coba kali ini rata-rata adalah besar, artinya waktu yang dipergunakan untuk mendownload halaman web ketika menggunakan content cache jauh lebih kecil, dibandingkan dengan ketika tanpa menggunakan content cache. Meskipun pada beberapa kasus, terjadi penyimpangan pola, dimana waktu yang diperlukan untuk mendownload ketika menggunakan content cache ternyata lebih besar, dibandingkan dengan ketika tidak menggunakan content cache.

Hal ini dapat disebabkan karena objek Internet yang dijadikan objek uji coba tidak dapat dicache, dan kondisi lalu lintas data tidak memungkinkan untuk

menempuh waktu yang lebih singkat. Sehingga ketika objek tersebut didownload untuk kedua kalinya, waktu yang dibutuhkan justru lebih besar.

Pada uji coba kali ini grafik yang dihasilkan adalah sebagai berikut :



Gambar 5.2 grafik pada uji coba kedua

Pada grafik ini perbedaan waktu yang dipergunakan untuk download halaman web pada kedua kondisi terlihat jelas.

Pada uji coba ketiga, yaitu ketika klien melakukan request terhadap hyperlink yang terdapat pada level 2, terjadi lagi perbedaan pola selisih waktu akses yang dibutuhkan untuk mendownload halaman web. Berdasarkan perhitungan matematis yang telah diterapkan pada level 1 dengan persamaan (1):

Perbedaan waktu rata-rata sebelum menggunakan content cache dan sesudah menggunakannya dari uji coba ini adalah sebagai berikut :

$\Sigma(t2\text{-}t1) = (13564 - 1241) + (8865 - 8865) + (70033 - 2) + (1051 - 0) +$

$(112 - 9127) + (17906 - 0) + (4516 - 4) + (2222 - 0) + (249 - 0) +$

$(890 - 26911) + (8457 - 30442) + (108319 - 12) + (58090 - 5465) +$

$(9497 - 0) + (3359 - 2893)$

$= 222168$

$n = 15$

maka,

$$\Lambda = \frac{222168}{15}$$

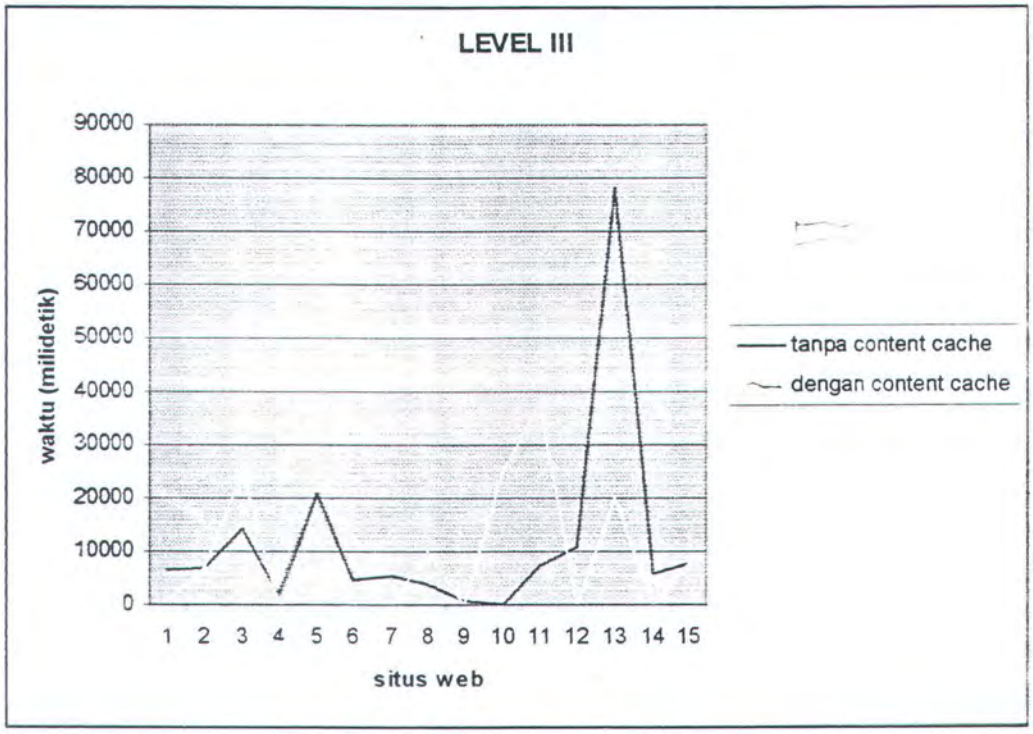
$\Lambda = 14811{,}2$. Nilai ini adalah bilangan terkecil dari ketiga rangkaian percobaan yang dilakukan terhadap situs-situs web yang sama.

Tabel 5.3 Level 3

No	Alamat	Tanpa Content Cache (dalam mili detik)	Dengan Content Cache (dalam mili detik)
1	www.whitehouse.gov	6443	859
2	www.cinema.com	7016	7016
3	www.bapepam.go.id	14095	25003
4	www.cpan.com	1763	0
5	www.depdagri.go.id	20732	2196
6	www.cia.gov	4730	0
7	www.danamon.co.id	5516	4
8	www.deplu.go.id	3969	10019
9	www.assunnah.or.id	842	7
10	www.bes.com	4	24901
11	www.alsofwah.or.id	7448	37785
12	www.at-taqwa.com	10665	9
13	www.reuters.com	78096	20032
14	www.islamonline.com	5871	0
15	www.liputan6.com	7871	7871

Kendatipun demikian penggunaan content cache masih memberikan keuntungan selisih waktu rata-rata untuk mendownload isi suatu halaman web.

Grafik hubungan antara waktu total download dengan situs web terkait dari uji coba ketiga ini adalah sebagai berikut :



Gambar 5.3 grafik pada uji coba ketiga

Perhitungan dari percobaan ini adalah sebagai berikut :

$\Sigma(t2\text{-}t1) = (6443 - 859) + (7016 - 7016) + (14095 - 25003) - (1763 - 0) +$

$(20732 - 2196) + (4730 - 0) + (5516 - 4) + (3969 - 10019) +$

$(842 - 7) + (4 - 24901) + (7448 - 37785) + (10665 - 9) +$

$(78096 - 20032) + (5871 - 0) + (7871 - 7871)$

$=39359$

dimana,

n = 15

maka,

$\Lambda = 2623.933$

$$\Lambda = \frac{39359}{15}$$

BAB VI

KESIMPULAN DAN SARAN

BAB VI

KESIMPULAN DAN SARAN

Setelah melakukan beberapa perancangan, implementasi dan pengujian, maka Tugas Akhir ini mencapai tahap pengambilan kesimpulan disertai pula dengan saran bagi para peminat yang ingin menyempurnakan program ini

6.1 KESIMPULAN

Kesimpulan yang dapat ditarik dari rangkaian Tugas Akhir ini adalah sebagai berikut :

1. Penggunaan Server Proxy Content Cache dapat meningkatkan kecepatan akses pada Internet.
2. Beberapa objek Internet tidak dapat dicache (noncacheable), sehingga mengakibatkan tidak terjadi peningkatan kecepatan ketika mempergunakan program ini.
3. Dari uji coba yang dilakukan, didapat bahwa dengan program ini hasil optimal dicapai pada request *hyperlink* level kedua

6.2 SARAN

Terdapat beberapa hal yang perlu dibenahi dalam program dibuat oleh penulis ini. Untuk mempermudah dalam mempelajari dan berguna sebagai bahan pertimbangan bagi mereka yang berminat pada program ini, penulis memberikan saran-saran umum sebagai berikut :

1. Perlu adanya perbaikan antar muka, agar lebih mudah dipergunakan

2. Perlu adanya file laporan (file log), sehingga apabila terjadi kesalahan mudah dalam merunutnya
3. Adanya opsi / pilihan untuk level yang berbeda-beda
4. Perlu adanya dokumentasi program untuk dipelajari oleh orang lain yang memiliki minat pada pengembangan program ini

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[ASN-00] Server Linux, Ahmad Sofyan, 2000

[ASP-02] Pemrograman Berbasis Linux, Andry Syahputra, 2002

[BFI-98] Berners-Lee,R. Fielding, U.C. Irvine, L. Masinter, rfc 2396 Xerox Corporation, August 1998

[CNT-99] Computer Network, Tanenbaum, Prentice Hall, 1999

[DWC-97] Request For Comment 2186, Duanne Wessels & Claffy, National Laboratory for Advanced Network Research, September 1997

[DWS-00] Duanne Wessels, www.squid-cache.org, 2000

[EKR-89] Request For Comment 1118, E. Krol, University of Illinois Urbana, September 1989

[JMW-71] Request For Comment 147, The Definition of Socket, Joel.M.Winett, 7 May 1971

[JPM-03] jupitermedia corporations www.webopedia.com, 2003

[LPM-95] Linux Programmer Manual, June 1995

[NWG-99] Request For Comment 2616, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Network Working Group, June 1999

[OWP-01] Linux Untuk Warung Internet, Onno W Purbo, 2001

[PEB-03] Paul E. Black, National Institutes of Standard Technologies 2003

[RCL-00] Robert Collins, SQU, 5 November 2000

[REI-02] Administrasi Sistem Linux Redhat, Richardus Eko Indrajit, dkk, 2002

[RLR-91] Ronald L Rivest, Prof. rfc 1321, RSA Data Security, 1991

[RNL-01] Ronald, squid-user, Maret 2001

[RLS-01] Ralph Swick, www.w3.org, 2001

[TBL-94] Tim Berners Lee, URL spec, http://www.w3.org/Addressing/URL/url-spec.txt21, Maret 1994

[WCU-00] Programming in C, System Calls and Subroutines Using C, www.cs.cf.ac.uk, A. David Marshall, 29 Maret 1999

[WMB-95] www.wombat.doc.ic.ac.uk, 1995

[WMB-98] www.wombat.doc.ic.ac.uk, 1998

[WPH-01] www.programmers-heaven.com, 2001

[WSO-00] Squid a User Guide, www.squid-cache.org, Oskar Pearson, Qualica Technologies, 2003