WEB PROXY

Oleh: **Abel Pires da Silva* (T 101 95 0563)**

Pembimbing I: **Ir. Sudjadi, M.T.** Pembimbing II: **Agung Budi P., ST., M IT.**

ABSTRAK

Web Proxy adalah suatu program yang diletakkan di antara suatu komputer/jaringan komputer dengan Internet. Web Proxy bertindak sebagai perantara (middleman) bagi web browser dan web server. Web Proxy bertindak sebagai server bagi web browser dan proxy bertindak sebagai client bagi web server.

Tugas Akhir ini membuat Web Proxy dengan menggunakan pemrograman socket, ditulis dengan bahasa pemrograman C dengan fasilitas GNU C Compiler dan berjalan pada Sistem Operasi Linux.

Web proxy ini dapat dijadikan acuan untuk mempelajari pembuatan sistem client server dan juga pemrograman jaringan terutama pengembangan fungsi-fungsi proxy yang lebih rumit

I. LATAR BELAKANG

Komunikasi di dalam Internet antara suatu web browser (client) dengan suatu web server (server) umumnya dilakukan secara langsung dengan menggunakan paradigma hubungan client/server.

Seiring dengan perkembangan teknologi Internet, dan meningkatnya resiko keamanan suatu komputer/jaringan komputer yang terhubung langsung dengan Internet, maka dikembangkanlah suatu teknologi dalam jaringan komputer yang mampu bertindak sebagai perantara antara suatu komputer/jaringan komputer dengan Internet, yang dikenal sebagai *proxy*.

Proxy adalah suatu program yang diletakkan di antara suatu komputer/jaringan komputer dengan Internet. *Proxy* bertindak sebagai perantara (*middleman)* bagi kedua pihak.

Proxy bertindak sebagai *server* bagi pengakses Internet (*web browser*) dan *proxy* bertindak sebagai *client* bagi *web server*.

II. LANDASAN TEORI

2.1. Pendahuluan

Dalam melakukan komunikasi data pada jaringan komputer, protokol memegang peranan untuk mengatur suatu komputer berkomunikasi dengan komputer lainnya. Untuk berkomunikasi, protokol yang dipakai haruslah sama, karena protokol, mirip dengan bahasa, sehingga komputer perlu “berbicara” dalam bahasa yang sama jika hendak berkomunikasi.

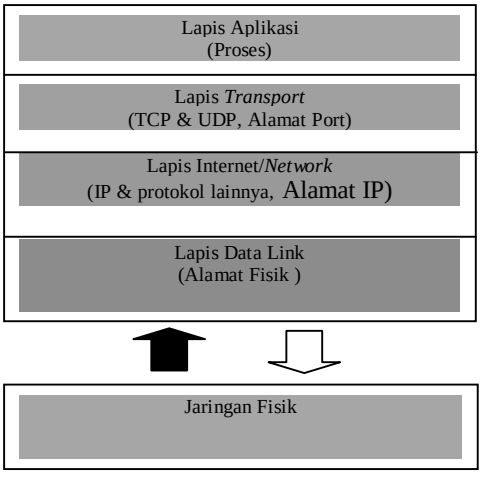
Terdapat beragam jenis protokol yang ada dewasa ini, namun protokol TCP/IP (*Transmission Control Protocol/Internet Protocol*) merupakan protokol yang paling banyak dipakai saat ini. TCP/IP adalah sekelompok protokol yang mengatur komunikasi data komputer di internet. Sepanjang menggunakan protokol TCP/IP, maka perbedaan jenis komputer maupun Sistem Operasi yang dipakai tidak menjadi masalah.

2.2 Protokol TCP/IP

TCP/IP adalah sekumpulan protokol yang didesain untuk melakukan fungsi-fungsi komunikasi data. TCP/IP terdiri dari sekumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu dari komunikasi data. Setiap protokol memiliki tugas yang jelas. Protokol yang satu tidak perlu mengetahui cara kerja protokol yang lain selama pengiriman dan penerimaan data dilakukan.

Prinsip kerja yang demikian menjadikan protokol TCP/IP sebagai protokol yang fleksibel karena tidak spesifik terhadap satu komputer atau peralatan jaringan tertentu, sehingga dapat diterapkan dengan mudah disetiap jenis komputer dan antarmuka jaringan. Agar TCP/IP dapat berjalan diatas antarmuka jaringan tertentu, yang perlu dilakukan adalah merubah protokol yang berhubungan dengan antarmuka jaringan yang bersangkutan saja.

Kumpulan protokol TCP/IP ini dapat dimodelkan dengan empat lapis TCP/IP dan pengalamatannya seperti Gambar 2.1



Gambar 2.1 Hubungan antar lapis TCP/IP dan alamatnya

2.2 Koneksi TCP

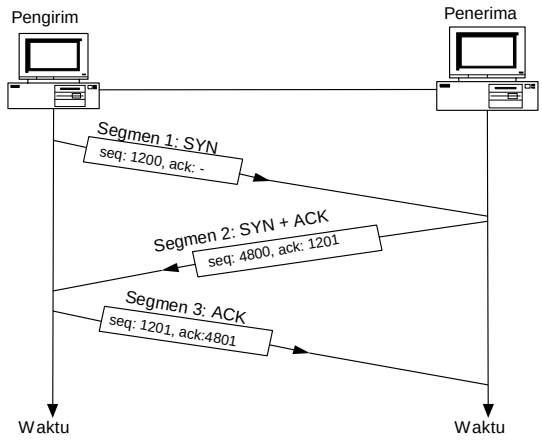
TCP merupakan protokol *connection oriented*. Protokol berorientasi koneksi ini (*Connection oriented protocol*) membangun suatu jalur *virtual* antara sisi sumber dan sisi tujuan. Semua *segmen* dari pesan dikirimkan melalui jalur *virtual* ini.

Dalam TCP proses transmisi data dilakukan melalui 2 prosedur, yaitu: **pembentukan koneksi** dan **pemutusan koneksi**.

a. Pembentukan koneksi

TCP mentransmisi data dalam bentuk *full-duplex*, artinya jika dua *host* terkoneksi, maka keduanya dapat saling mengirimkan *segmen* secara serentak. Sebelum dilakukan transfer data, maka kedua sisi harus memulai komunikasi dan untuk itu perlu satu sisi perlu mendapatkan persetujuan dari sisi yang lain untuk melakukan pertukaran data.

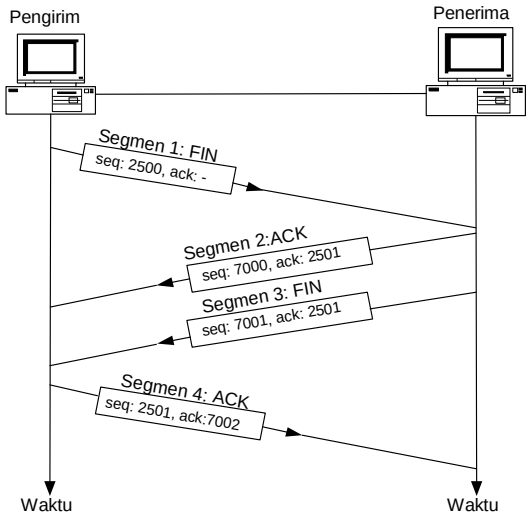
b. *Three-way Hanshake* (Tiga jalan jabat tangan)



Gambar 2.5 Pembentukan koneksi (Tiga jalan jabat tangan)

c. Pemutusan koneksi pada TCP

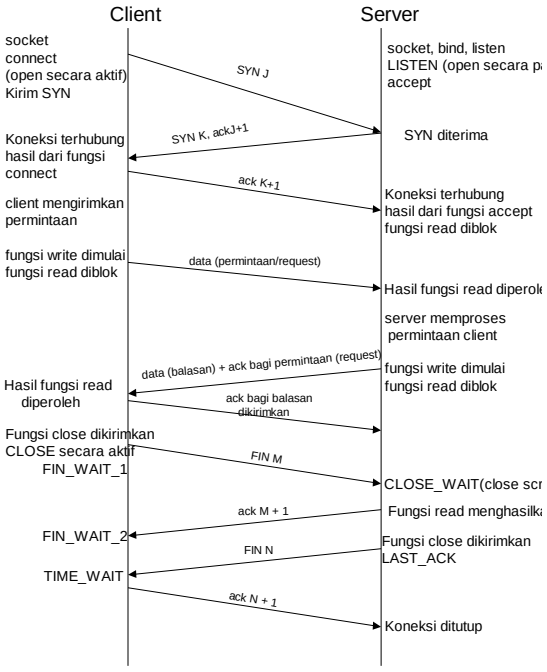
Jika pada proses pembentukan koneksi hanya dibutuhkan 3 buah *segmen*, dalam proses pemutusan koneksi dibutuhkan 4 buah *segmen*, dengan urutan sebagai berikut (Gambar 2.6 menunjukkan proses ini):



Gambar 2.6 Proses pemutusan koneksi TCP

2.3 Pertukaran Paket pada TCP

Gambar 2.8 menunjukkan pertukaran paket aktual yang terjadi pada koneksi TCP yang berturut-turut: pembentukan koneksi, transfer data, dan pemutusan koneksi.



Gambar 2.7 Pertukaran paket aktual pada koneksi TCP

2.4 *Proxy*

Proxy merupakan jenis *server* dan *client* sekaligus, yaitu *proxy* sebagai *server* bagi *web-browser* dan *proxy* sebagai *client* bagi *web-server*. Urutan pengerjaan *proxy* dilakukan sebagai berikut:

a. Pertama, membuat/menulis *server* yang akan menangani permintaan dari *web-browser*. *Server* inilah yang mengembalikan halaman-halaman HTML ke *web-browser*
b. Kedua, menulis/membuat program bagi sisi *client*, yang melakukan koneksi ke *web-server*
c. Ketiga, mengkombinasi kedua sisi diatas untuk membentuk suatu *web-proxy* sederhana.

Server merupakan program yang menunggu permintaan koneksi pada suatu *port* tertentu.

Untuk membuat *server* ini dibutuhkan sistem pemanggilan antara lain: *socket*, *bind*, *listen*, *accept*, *read*, *write*, dan *close*.

Client melihat alamat IP *server* kemudian membuat *socket* dan memanggil *connect* untuk membangun koneksi, setelah koneksi terbentuk, barulah proses pertukaran data dilakukan dengan menggunakan fungsi-fungsi *read* dan *write* pada *socket*.

Sistem pemanggilan yang ada untuk *client* adalah: *socket*, *connect*, *read*, *write*, dan *close*.

Gabungan kedua sisi diatas akan membentuk suatu *web-proxy*.

III. DISAIN *WEB-PROXY*

3.1 Pendahuluan

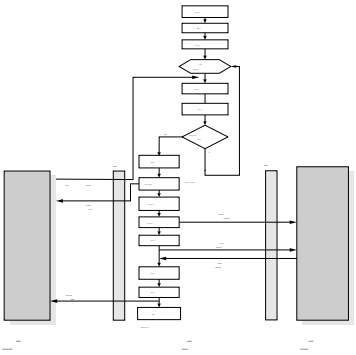
Arsitektur *web-proxy* ini disusun dari 3 (tiga) bagian utama:

1. Sisi *server* dari *proxy*
2. Bagian filter
3. Sisi *client* dari *proxy*

Ketiga komponen di atas yang menjalankan fungsi dari *proxy*. *Proxy* didisain untuk mampu menangani banyak proses (*multi process*) yaitu dengan memanggil fungsi *fork*. Fungsi *fork* ini jika berhasil akan menciptakan satu proses anak (*child process*) yang identik dengan proses *parent* sebelumnya (artinya mewarisi semua sifat dari proses *parent*).

Setelah *fork()* berhasil, yaitu dengan mengecek nilai balik tidak sama dengan nol (*fork* != 0) proses *parent* akan mendengarkan kembali koneksi dari *web browser (listening).* Proses *child* akan mengambil alih koneksi dari *web browser*, dan akan menangani proses pertukaran data dari *web browser* dan menuju ke *web server*

Gambar 3.1 adalah struktur lengkap *web proxy* yang tersusun dari 3 komponen di atas dengan *port* tempat *proxy* dan *web server* berjalan.



Gambar 3.1 Arsitektur *web-proxy*

IV. PENGUJIAN DAN ANALISA PROGRAM

Dalam Bab ini, terdapat 2 (dua) kemampuan program yang akan diuji, yaitu pertama, kemampuan program untuk menangani *multi client* dan kemampuan program untuk melakukan filter alamat yang dikirimkan oleh *web browser*.

4.1 Pengujian kemampuan menangani *multi client.*

Untuk melakukan pengujian ini, program diaktifkan pada *port* 80, untuk itu diperlukan akses masuk sebagai *root* pada sistem operasi Linux. Perintah untuk mengaktifkan program ini adalah: #./*proxy* 80

Kemudian perlu dilihat apakah *proxy* sudah berjalan (aktif) mendengarkan pada nomor *port* tersebut dengan perintah #*netstat –a –t.* Perintah ini akan menghasilkan suatu daftar *socket* TCP yang sedang mendengarkan koneksi beserta nomor *port*-nya. Jika *proxy* dengan nomor *port* 80 serta *apache web server* dengan *port* 8000 terdapat pada daftar tersebut maka langkah selanjutnya adalah mengaktifkan 4 (empat) jendela *web-browser* (sebagai *client*) yang akan mengirimkan koneksi secara hampir bersamaan ke *Apache web server*.

Dari hasil pengujian terbukti bahwa keempat jendela *web browser* yang mengakses halaman *apache web server* berhasil menerima respon dari *apache web server* pada waktu yang hampir bersamaan. Hal ini membuktikan kemampuan *proxy* untuk melakukan *multi process* dengan fungsi *fork()*.

4.2. Pengujian Kemampuan filter alamat dari program

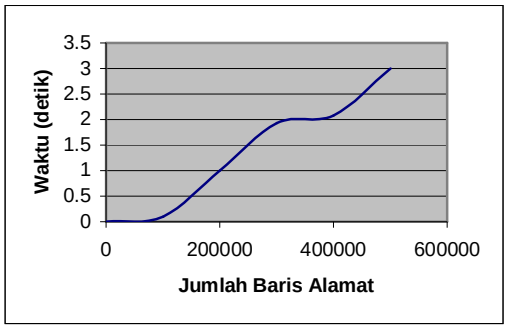
Pada pengujian program ini, program diakses menggunakan *web browser* yang sengaja memanggil alamat-alamat yang terdapat pada *file* check.dat, yang artinya akan ditolak oleh program.

Terdapat 6 (enam) alamat yang sengaja ditempatkan pada baris pertama pada *file*, baris ke-100.000, baris ke-200.000, baris ke-300.000, baris ke-400.000 dan baris ke-500.000.

Tiap alamat di atas yang diakses akan ditolak oleh program dengan menuliskan pesan: “ERROR 403:

FORBIDDEN, ACCESS DENIED!" pada *web browser*.

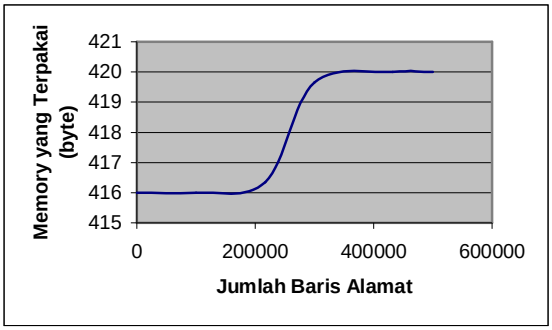
Dari hasil pengujian, terdapat hubungan antara jumlah baris pada *file* check.dat dengan waktu yang dibutuhkan oleh program untuk melakukan perbandingan. Hal ini terlihat pada Gambar 4.1 di bawah ini.



Gambar 4.1 Grafik hubungan antara jumlah baris pada *file* dengan waktu

Dari grafik terlihat bahwa terdapat hubungan antara waktu dan jumlah baris, yaitu semakin banyak jumlah baris yang terdapat pada *file* check.dat yang perlu diuji, maka waktu yang diperlukan untuk pengujian tersebut pun akan semakin lama.

Dari pengujian juga diperoleh hubungan antara memori yang terpakai oleh program berbeda-beda untuk jumlah baris pada *file* check.dat yang berlainan. Gambar 4.2 di bawah ini menunjukkan hal ini.



Gambar 4.2 Grafik hubungan antara jumlah baris alamat dengan memori yang terpakai

Dari grafik di atas terlihat bahwa peningkatan jumlah baris alamat pada *file* check.dat tidak menyebabkan peningkatan pemakaian memori yang berarti (signifikan).

Dalam kedua macam pengujian di atas, *proxy* menggunakan metode baca/tulis (*read*/*write*) secara *blocking* yang berarti *proxy* melakukan proses baca/tulis hanya sekali untuk masing-masing proses, sampai proses baca/tulis tersebut selesai dilakukan. Hal ini sangatlah riskan, karena dengan demikian, *proxy* tidak akan dapat melakukan proses baca/tulis lebih dari sekali, artinya untuk proses baca dari *web* browser maupun *web server*, dimana pesan yang diterima terdiri dari beberapa kali (lebih dari satu) operasi tulis oleh *web* browser maupun *web server*, maka *proxy* hanya dapat membaca (menerima pesan) pesan dari operasi tulis yang pertama saja, pesan dari operasi tulis selanjutnya tidak akan dibaca oleh *proxy*. Selain itu, jika data/pesan yang akan dibaca oleh *proxy* lebih besar dari ukuran *buffer* yang dipakai untuk menyimpan hasil pembacaan, maka *proxy* hanya dapat membaca pesan tersebut hingga sebesar ukuran *buffer* saja, sisa pesan lainnya tidak akan terbaca oleh *proxy*, karena itu, maka ukuran *buffer* pada *proxy* perlu ditetapkan sebesar mungkin.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil perancangan dan implementasi serta analisa, maka dapat diambil kesimpulan sebagai berikut:

1. *Web proxy* berhasil diuji dalam komunikasi *client*/*server* pada mesin/komputer lokal.
2. Program *web proxy* berhasil menjalankan fungsinya untuk menangani *multi process*,

yaitu kemampuan untuk melayani 4 (empat) *client* pada saat yang bersamaan.

3. Program *web proxy* berhasil melakukan filter terhadap alamat terlarang yang hendak diakses oleh *web browser*.
4. Semakin banyak jumlah baris alamat yang hendak difilter yang terdapat pada *file* check.dat maka waktu (dalam detik) yang diperlukan untuk melakukan filterpun akan semakin lama.
5. Semakin banyak jumlah baris alamat pada *file* check.dat tidak terlalu menyebabkan kenaikan pemakaian memori yang berarti (tidak signifikan).

Saran

Berdasarkan perancangan, hasil pengujian dan analisa maka dapat disarankan hal-hal berikut untuk peningkatan/perbaikan, antara lain:

1. Proses *write()* dan *read()* pada diskriptor *socket-socket* sebaiknya menggunakan fungsi *non-blocking socket* sehingga dapat mengurangi kegagalan program dalam proses baca dan tulis pada *socket*. Selain itu kontrol program terhadap *socket* mana yang siap untuk dibaca maupun yang siap untuk ditulisi akan semakin baik. Fungsi-fungsi yang biasa digunakan untuk tujuan *non-blocking socket* ini seperti fungsi *thread, poll* maupun *select.*
2. Untuk implementasi program secara nyata, maka perlu dilakukan penguraian *request* HTTP dari *web browser* untuk memperoleh alamat yang diminta oleh *browser* tersebut.
3. Perlu diperhatikan standar permintaan dan respon pada protokol HTTP, sehingga respon penolakan dari program *web proxy* dapat dilakukan berdasarkan standar-standar tersebut, dan tidak perlu melakukan penulisan penolakan "paksa" terhadap *web browser*, karena jenis respon yang tidak sesuai standar protokol HTTP seperti ini tidak lagi dipakai oleh versi protokol HTTP diatas versi HTTP/1.0
4. Program dapat dikembangkan untuk paradigma *Volume Based Internet Charging*, yaitu pengenaan biaya akses Internet berdasarkan banyaknya jumlah data yang dikirim atau diterima oleh pengguna akses Internet.
5. Kelambatan proses yang terjadi akibat banyaknya jumlah baris alamat yang akan dilakukan perbandingan perlu dicari penyebabnya, sehingga dapat diperoleh suatu program yang bekerja dengan cepat.

VI. DAFTAR PUSTAKA

1. Stevens, Richard W., UNIX Network Programming: Networking APIs *Sockets* and XTI, Volume 1, Prentice Hall, 1998.
2. Forouzan, Behrouz A., TCP/IP Protocol Suite, McGraw-Hill, Inc., 2000.
3. Donahoo, Michael J. & Calvert, Kenneth L., The Pocket Guide to TCP/IP *Sockets*, C Version, Morgan Kaufmann Publisher, 2001.
4. Comer, Douglas, Internetworking With TCP/IP: Principles, Protocols, and Architectures, Volume 1, Prentice Hall Englewood Cliff-New Jersey, 1988.
5. Wierer, Jeff, *Proxy* Gods FAQ, MVP & Svyatoslav Pidgorny, 2002.
6. Kernighan, Brian W. & Ritchie, Dennis M., The C Programming Laguage, 2nd Edition, Prentice-Hall Inc., 1988.

7. Hartono, Jogiyanto, Konsep Dasar Pemrogaman Bahasa C, Edisi kedua, ANDI Yogyakarta, 2000.
8. W. Purbo, Onno, TCP/IP: Standar, Desain dan Implementasi, PT. Elex Media Komputindo, 1998.

***Abel Pires da Silva**, Mahasiswa Teknik Elektro angkatan 1995, Lahir di Timor-Leste 24 Mei 1976, menyelesaikan pendidikan SMA di SMN2 Dili, Selama masa cuti kuliah (1999-2002) pernah bekerja sebagai Wartawan, Aktivis Forum Nasional NGO (LSM) Timor-Leste, dan terakhir bekerja pada ILO (*International Labour Organization*) Kantor Perwakilan Dili.

Menyetujui,
Dosen Pembimbing II

Agung Budi Prasetijo, ST,MIT.