

Implementasi dan Analisa Penggunaan Protokol SCTP pada Aplikasi Berbasis Protokol HTTP

Eriefian Addinata¹, Tri Brotoharsono², Bayu Erfianto³

^{1,2,3}Fakultas Teknik Informatika Institut Teknologi Telkom, Bandung

Jl. Telekomunikasi Dayeuhkolot Bandung 40257, Indonesia

¹addie.bae@gmail.com, ²tbh@ittelkom.ac.id, ³erf@ittelkom.ac.id

Abstrak

Protokol HTTP versi 1.1 merupakan versi terbaru dari protokol HTTP yang digunakan oleh aplikasi web server dan web client untuk melakukan transaksi HTTP. Karakteristik dari HTTP 1.1 adalah mampu melayani transaksi secara *persistent* dan mendukung konsep *pipeline*. Protokol ini pada umumnya menggunakan protokol TCP sebagai protokol transportnya. Transaksi HTTP yang dilakukan secara *pipeline* akan dikirimkan oleh TCP secara terurut sesuai dengan karakteristiknya. Hal ini dapat menyebabkan kondisi *HOL blocking* dikarenakan TCP tidak bisa memisahkan pesan-pesan HTTP yang dapat berdiri sendiri. Penggunaan protokol SCTP sebagai protokol transport pada aplikasi HTTP dapat mengurangi kondisi *HOL blocking* dengan memanfaatkan kemampuan *multistreaming* yang ada pada protokol SCTP. Performansi SCTP dengan menggunakan stack protokol LKSCTP dilihat dari sisi page response time yang dihasilkan pada kondisi jaringan tanpa adanya paket data yang hilang cenderung lebih lambat dan menghasilkan throughput yang lebih buruk. Namun untuk kondisi jaringan yang terdapat paket data yang hilang maka page response time yang dihasilkan akan lebih cepat dan menghasilkan throughput yang lebih baik jika dibandingkan dengan penggunaan protokol TCP.

Kata kunci : HTTP, persistent connection, pipeline, TCP, HOL blocking, SCTP, multistreaming

Abstract

HTTP protocol version 1.1 is newer version of HTTP protocol which used by web server and web client application for doing HTTP transaction. Characteristic of HTTP 1.1 is able to handle transaction persistently and support pipelining concept. This protocol generally use TCP as a transport protocol. Pipelined HTTP transaction will be delivered by TCP ordered which is its characteristic. This case can cause HOL blocking condition because TCP couldn't separate HTTP messages independently. Using SCTP as a transport protocol for HTTP application can reduces HOL blocking with use of multistreaming ability from SCTP protocol. Performance result of SCTP using LKSCTP protokol stack in page response time side under network with no loss packet is affectively slower and also produce worse throughput. But under network with loss packet will produce faster page response time and also a better throughput than TCP protocol.

Keywords: HTTP, persistent connection, pipeline, TCP, HOL blocking, SCTP, multistreaming

1. Pendahuluan

1.1 Latar Belakang

Pada arsitektur jaringan TCP/IP atau yang lebih dikenal dengan jaringan Internet, aplikasi client-server yang berbasis protokol HTTP pada umumnya menggunakan protokol TCP sebagai protokol lapisan transportnya untuk melakukan komunikasi antara web client dan web server. Hal ini dikarenakan protokol TCP sudah dikenal luas dan terutama juga karena dirancang untuk menyediakan layanan pengiriman data yang handal dengan end-to-end deteksi dan koreksi kesalahan yang dibutuhkan oleh protokol HTTP. Namun aplikasi yang berbasis HTTP tidak dibatasi untuk berjalan hanya pada protokol TCP, dengan kata lain memungkinkan juga aplikasi tersebut untuk dapat berjalan pada protokol lainnya asalkan protokol tersebut mampu menyediakan layanan pengiriman data yang handal.

Dalam perkembangan dari arsitektur jaringan TCP/IP selanjutnya, pada lapisan transport terdapat protokol SCTP yang merupakan protokol baru yang karakteristiknya juga mampu menyediakan layanan yang handal bagi protokol lapisan aplikasi yang akan menggunakannya sebagai protokol transport. Motivasi dari adanya protokol SCTP adalah karena keterbatasan dari protokol TCP sebagai protokol yang menyediakan layanan handal. Sejumlah aplikasi baru, yang mendapatkan keterbatasan pada protokol TCP, menyatukan protokol pengiriman datanya sendiri yang handal diatas protokol UDP. Hal ini dikarenakan karakteristik TCP yang bersifat *reliable* dan juga tegas dalam hal pengurutan dari data-data yang dikirimkan namun beberapa aplikasi hanya membutuhkan pengiriman yang handal tanpa pemeliharaan urutan sedangkan yang lainnya akan tercukupi hanya dengan pengurutan sebagian dari datanya sehingga kasus *head-of-line blocking* yang

ada pada TCP menyebabkan waktu penundaan yang tidak perlu.

Kondisi *head-of-line (HOL) blocking* yang terjadi dalam implementasi HTTP melalui TCP merupakan suatu kondisi yang terjadi karena TCP tidak bisa memisahkan response-response HTTP yang berdiri sendiri secara lebih logis di dalam mekanisme pengirimannya terutama pada HTTP versi 1.1 yang mendukung konsep *persistent* dan *pipeline*. Pada protokol TCP, response-response HTTP tersebut dikirimkan terurut secara serial melalui bytestream tunggal. Jika TPDU ke- i yang berisi response HTTP ke- i hilang dalam jaringan, maka TPDU ke- $i+1$ yang tiba diluar urutan tidak akan dikirimkan ke web client sampai TPDU yang hilang dikirim ulang dan diterima. Apabila TPDU ke- $i+1$ membawa bagian dari objek web lainnya (response ke $i+n$, $n \geq 1$), hal ini akan menghalangi pengiriman bagian objek web ke- $i+n$ tersebut ke web client.

Protokol SCTP mempunyai kelebihan dalam permasalahan yang ada pada protokol TCP tersebut. Di dalam protokol SCTP terdapat layanan *multistreaming* yang merupakan kemampuan dari protokol SCTP untuk bisa membuka banyak stream dalam sebuah asosiasi (sebuah koneksi dalam TCP). Stream-stream tersebut bersifat mandiri tetapi tetap berkaitan. Ketika sebuah stream terhalang (misal, TPDU yang menunggu dikarenakan ada proses pengiriman ulang terhadap TPDU yang hilang sebelumnya) maka tidak akan mengganggu stream yang lainnya. HTTP server melalui SCTP (dalam hal ini menggunakan mekanisme *multistreaming*) akan menyediakan interaktivitas yang lebih baik karena setiap request akan dilayani menggunakan stream-stream berbeda yang berdiri sendiri untuk tiap asosiasi.

1.2 Perumusan Masalah

hal-hal yang akan dilakukan dalam tugas akhir ini, antara lain:

1. Melakukan implementasi pada aplikasi HTTP yang telah ada sehingga bisa menggunakan protokol SCTP sebagai protokol lapisan transportnya dengan memanfaatkan modul protokol SCTP yang telah ada pada sistem operasi.
2. Mengukur performansi HTTP server yang menggunakan SCTP sebagai protokol lapisan transportnya dilihat dari page response time dan throughputnya.

1.3 Batasan Masalah

Adapun batasan-batasan dari permasalahan dalam tugas akhir ini adalah:

1. Implementasi hanya dilakukan pada sisi server dengan melakukan modifikasi pada web server Engine X.

2. Sistem operasi yang digunakan adalah GNU/Linux yang telah ditambahkan modul protokol SCTP (LKSCPT) pada kernelnya.
3. Analisa yang dilakukan hanya pada protokol HTTP 1.1 dan untuk kondisi web statis.

1.4 Tujuan

Tujuan dibuatnya tugas akhir ini antara lain adalah:

1. Mengimplementasikan penggunaan protokol SCTP pada aplikasi server yang berbasis protokol HTTP.
2. Menganalisa pengaruh protokol SCTP pada web server dilihat dari sisi page response time dan throughput yang dihasilkan.

1.5 Metodologi Penyelesaian Masalah

Metodologi yang digunakan dalam penelitian tugas akhir ini meliputi hal-hal sebagai berikut:

1. Pengumpulan data dan studi literatur

Pada tahap ini akan dilakukan pemahaman terhadap protokol SCTP, HTTP, dan TCP. Selain itu juga dilakukan pemahaman terhadap SCTP socket API pada LKSCPT (Linux Kernel SCTP).

2. Perancangan dan implementasi

Dalam tahap ini akan dilakukan perancangan penggunaan fase-fase pada protokol SCTP dan mengimplementasikannya ke dalam beberapa fungsi yang berkaitan pada kode sumber aplikasi web server Engine X.

3. Skenario pengujian

Dalam tahap ini akan dilakukan pengujian performansi terhadap hasil implementasi yang telah dilakukan dengan skenario kasus uji melakukan pengunduhan sebuah halaman web statis yang berisi sejumlah berkas gambar yang berbeda dan dilakukan terhadap beberapa variasi ukuran bandwidth jaringan. Perangkat lunak tambahan yang digunakan antara lain *phhttpget* sebagai web client dan disiplin trafik TC yang digunakan untuk simulasi bandwidth jaringan dengan bantuan aplikasi *netem* untuk melakukan simulasi delay dan paket loss.

4. Analisa performansi

Dalam tahap akan dilakukan analisa terhadap hasil pengujian yang dilakukan dengan mengamati grafik hasil dari pengujian dan membandingkannya untuk mengetahui perbedaan antara penggunaan TCP dan SCTP.

5. Pembuatan laporan

Merupakan tahap akhir yang mencakup penyelesaian penulisan buku dan membuat kesimpulan dari hasil penelitian tugas akhir ini.

2. Dasar Teori

2.1 Protokol HTTP

Protokol HTTP merupakan protokol lapisan aplikasi pada struktur jaringan TCP/IP yang digunakan oleh aplikasi web client dan web server dalam melakukan komunikasinya satu sama lain. Pada arsitektur jaringan TCP/IP atau yang lebih dikenal dengan jaringan Internet, aplikasi yang menggunakan protokol HTTP pada umumnya menggunakan TCP sebagai protokol end-to-end pada lapisan transportnya. Hal ini dikarenakan HTTP membutuhkan koneksi yang handal sebagai media pengiriman dan penerimaan pesan-pesan request atau responsenya. Pada versi 1.0 dari protokol HTTP, karakteristik dari protokol tersebut adalah membuat satu koneksi TCP untuk setiap satu proses request-response. Untuk setiap pesan request yang dikirimkan oleh client maka akan terbentuk satu koneksi TCP antara client dan server yang kemudian digunakan untuk proses pengiriman pesan response oleh server melalui koneksi TCP yang telah terbentuk tersebut. Dalam perkembangannya, protokol HTTP versi 1.1 memiliki kemampuan tambahan yang merupakan perbaikan dari versi sebelumnya yaitu: *persistent connection* dan *pipeline*. [1][8]

1. Persistent Connection

Merupakan kemampuan dari protokol HTTP versi 1.1 dimana client dan server bisa melakukan pengiriman pesan-pesan request atau response secara berulang dalam satu koneksi TCP yang sama. [1][8]

2. HTTP pipelining

Mekanisme pipelining pada protokol HTTP adalah kemampuan untuk mengirimkan banyak pesan request HTTP dalam sebuah koneksi TCP yang terbentuk tanpa harus menunggu pesan response untuk masing-masing pesan request tersebut terlebih dahulu. [1][8]

2.2 Protokol TCP

Protokol TCP termasuk ke dalam lapisan transport pada struktur jaringan TCP/IP yang bersifat reliable, berorientasi koneksi, dan memberikan layanan pengiriman data berbasis bytestream. Protokol ini memberikan jaminan yang handal dalam melakukan proses pengiriman terhadap setiap data yang dikirimkan. Selain itu, TCP juga bersifat *ordered* yakni proses pengiriman data-datanya (TPDU) dilakukan secara berurutan. Apabila dalam proses pengiriman ada TPDU ke-*i* yang hilang di jaringan dan TPDU ke-*i*+1 tiba diluar urutan. Maka TPDU ke-*i*+1 yang tiba diluar urutan tersebut tidak akan dikirimkan ke protokol lapisan aplikasi sampai TPDU ke-*i* yang hilang dikirim ulang dan diterima oleh sisi penerima TCP. [5][7]

1. Koneksi TCP

Sebuah koneksi pada protokol TCP terdiri dari tiga fase yakni: fase pembukaan, fase pengiriman data, dan fase penutupan koneksi. Fase pembukaan koneksi menggunakan metode *3-way handshake* dan kemudian antara client dan server baru bisa saling mengirimkan data. Setelah proses pengiriman data selesai koneksi TCP akan diakhiri dengan fase penutupan. [8][9]

2.3 Protokol SCTP

Protokol SCTP termasuk dalam lapisan transport dalam struktur jaringan TCP/IP yang kedudukannya sama seperti protokol TCP. SCTP merupakan protokol yang bersifat reliable dan merupakan protokol message-oriented. Dalam proses pengirimannya, protokol SCTP mengirimkan data-data dari protokol aplikasi yang berada di atasnya ke dalam bentuk pesan-pesan yang dipisahkan menjadi *data chunk* dan *control chunk* yang masing-masing diidentifikasi oleh *chunk header*. Data chunk tersebut bisa dipecah-pecah lagi menjadi ukuran yang lebih kecil dan semuanya akan disatukan kembali menjadi pesan-pesan SCTP tunggal ketika sampai ke penerima. [4][9][10]

1. Asosiasi SCTP

Secara umum sebuah asosiasi SCTP terdiri dari tiga fase: fase pembentukan atau pembukaan asosiasi, fase pengiriman data, dan fase penutupan asosiasi. Tahap pembentukan asosiasi diawali oleh proses inisiasi menggunakan mekanisme *4-way handshake*. Setelah asosiasi terbentuk, antara client dan server baru dapat melakukan proses pengiriman data. Fase penutupan merupakan tahap terakhir dari sebuah asosiasi SCTP setelah proses pengiriman data selesai dilakukan sepenuhnya. [5][9][11]

2. Multistreaming

Multistreaming merupakan kemampuan dari protokol SCTP untuk mengirimkan pesan-pesannya ke dalam banyak stream yang berdiri sendiri namun tetap berada dalam satu kesatuan asosiasi tunggal. Masing-masing stream tersebut mempunyai urutan pengiriman datanya sendiri. [9][10][11]

2.4 Mekanisme transaksi HTTP melalui protokol SCTP

Implementasi aplikasi HTTP melalui protokol SCTP diterapkan dengan mengirimkan masing-masing pesan request HTTP ke dalam stream-stream yang berbeda. Dan kemudian menerima response-response HTTP yang berupa objek-objek web pada stream-stream yang berbeda. Stream-stream yang digunakan untuk melakukan transaksi HTTP tersebut tetap berada dalam satu asosiasi yang sama. Response HTTP yang dikirimkan oleh server

menuju client akan diterima oleh client pada stream yang sama dengan request HTTP untuk response tersebut dikirimkan. Dalam hal ini jumlah stream masukan dan stream keluaran yang digunakan harus sama. [6]

3. Perancangan

3.1 Analisa Sistem

1. Deskripsi umum aplikasi Engine X

Kondisi awal dari aplikasi Engine X yang menggunakan protokol TCP sebagai protokol transportnya sebelum dilakukan perubahan dapat dijelaskan sebagai berikut sesuai dengan tahapan dari protokol TCP:

a. Pembentukan koneksi

Fase pembentukan koneksi dari Engine X terdapat pada fungsi `ngx_open_listening_sockets`. Pembentukan pendeskripsi socket yang akan menggunakan protokol TCP sebagai protokol transport adalah dengan memberikan parameter `IPPROTO_TCP` ketika memanggil fungsi API socket, dan selanjutnya secara berurutan akan memanggil fungsi API `bind` dan `listen` untuk menyiapkan endpoint sebagai sebuah server yang akan menunggu koneksi dari client.

b. Transmisi data

Tahapan transmisi data terdiri dari dua bagian, yakni pengiriman pesan response HTTP dan pembacaan pesan request HTTP dari client.

➤ Pengiriman response HTTP ke client

Tahap pengiriman response HTTP terdapat pada beberapa fungsi antara lain: `ngx_linux_sendfile_chain` pada berkas `ngx_linux_sendfile_chain.c`, `ngx_writev_chain` pada berkas `ngx_writev_chain.c`, dan `ngx_unix_send` pada berkas `ngx_send.c`. Untuk fungsi `ngx_linux_sendfile_chain` akan menggunakan fungsi API `sendfile`, sedangkan fungsi `ngx_writev_chain` menggunakan API `writev` dan `ngx_send` akan menggunakan fungsi API `send` untuk mengirimkan pesan response-response HTTP nya ke client.

➤ Pembacaan request HTTP dari client

Pembacaan data yang berupa request HTTP dilakukan dengan menggunakan fungsi `ngx_unix_recv` pada berkas `ngx_recv.c`. Fungsi ini menggunakan API `recv` untuk membaca data yang berupa request HTTP dari client.

2. Deskripsi system

Sistem yang dibuat meliputi melakukan beberapa perubahan pada kode sumber aplikasi Engine X sehingga aplikasi HTTP server tersebut bisa menggunakan protokol SCTP sebagai protokol lapisan transportnya. Perubahan-perubahan yang

dilakukan sesuai dengan fase-fase dari protokol SCTP yang meliputi fase pembentukan asosiasi dan fase transmisi data.

a. Pembentukan Asosiasi

Tahapan pembentukan asosiasi terdapat pada fungsi `ngx_open_listening_sockets` yang ada di dalam berkas `ngx_connection.c`. Perubahan yang dilakukan adalah dimulai pada bagian pemanggilan fungsi pembentukan pendeskripsi socket menggunakan API socket (`ngx_socket`), yakni dengan menggunakan parameter protokol `IPPROTO_SCTP`. Selain itu juga melakukan inisiasi ukuran maksimal stream yang akan digunakan setelah pemanggilan fungsi API `bind`. Inisiasi jumlah maksimal stream dengan menggunakan fungsi API `setsockopt` pada pendeskripsi socket yang telah terbentuk sebelumnya dengan menggunakan parameter `SCTP_INITMSG` dan ukuran maksimal stream yang digunakan adalah 50. Ukuran ini disesuaikan dengan jumlah rata-rata objek web yang ada dalam satu halaman situs pada saat ini.

b. Transmisi data

Tahapan ini terbagi menjadi dua, yakni pengiriman pesan response HTTP dan pembacaan pesan request HTTP. Penjelasan selengkapnya adalah sebagai berikut:

➤ Pengiriman response HTTP

Tahap pengiriman pesan response HTTP berada pada fungsi `ngx_writev_chain_sctp` yang berada pada berkas `ngx_writev_chain.c` dan fungsi `ngx_unix_send_sctp` yang berada pada berkas `ngx_send.c` yang keduanya merupakan fungsi baru. Pada fungsi `ngx_writev_chain_sctp`, perubahan yang dilakukan adalah menggunakan fungsi API `sendmsg` yang berada pada fungsi baru yakni `writev_sctp`. Fungsi `writev_sctp` digunakan untuk membuat struktur variabel `msghdr` yang akan digunakan oleh fungsi `sendmsg` sebagai pesan SCTP yang akan dikirimkan. Selain itu untuk fungsi `ngx_unix_send_sctp` juga menggunakan fungsi `sendmsg` yang berada pada fungsi baru `ngx_sendmsg`. Fungsi `ngx_sendmsg` digunakan untuk membentuk struktur pesan `msghdr` yang akan dikirimkan sebagai pesan SCTP.

➤ Pembacaan request HTTP

Tahapan pembacaan pesan request HTTP berada pada fungsi `ngx_unix_recv_sctp` pada berkas `ngx_recv.c` yang merupakan fungsi baru. Fungsi ini menggunakan API `recvmsg` untuk membaca data dari pendeskripsi socket. Data yang dibaca akan ditampung ke dalam struktur `msghdr` yang telah didefinisikan pada awal fungsi.

Selain perubahan-perubahan yang telah disebutkan diatas, beberapa perubahan lainnya dapat dilihat pada tabel berikut ini:

Tabel 1: Daftar perubahan pada aplikasi Engine X

Nama fungsi / struktur variabel / berkas	Keterangan
a. struktur ngx_connection_s (pada berkas ngx_connection.h)	Pada struktur ini ditambahkan variabel stream_id yang digunakan untuk menampung nomor stream dari pesan request HTTP dan digunakan juga ketika akan mengirimkan pesan response HTTP.
b. berkas ngx_os.h	Di dalam berkas ini ditambahkan definisi dari fungsi-fungsi berikut: - ngx_unix_recv_sctp - ngx_writev_chain_sctp - writev_sctp - ngx_unix_send_sctp - ngx_sendmsg
c. struktur ngx_linux_io (pada berkas ngx_linux_init.c)	Isi struktur ini diubah menjadi: - ngx_unix_recv_sctp - ngx_readv_chain - ngx_udp_unix_recv - ngx_unix_send_sctp - ngx_writev_chain_sctp
d. fungsi ngx_http_set_keepalive (pada berkas ngx_http_request.c)	Dalam fungsi ini dilakukan perubahan untuk konfigurasi opsi socket menggunakan parameter SCTP_NODELAY.
e. berkas ngx_linux_config.h	Di dalam berkas ini ditambahkan pustaka protokol SCTP dari LKSCTP API: #include <netinet/sctp.h>

3.2 Tujuan Perancangan

Tujuan perancangan adalah untuk memberikan kemampuan kepada aplikasi HTTP server Engine X agar bisa menggunakan protokol SCTP sebagai protokol lapisan transportnya. Hasil yang diharapkan dari penerapan protokol SCTP ini adalah:

- Mengurangi atau mencegah kondisi HOL blocking yang terjadi ketika aplikasi HTTP server menggunakan protokol TCP sebagai protokol transportnya. Terutama ketika server mengirimkan response-response HTTP kepada client.
- Page response time yang dihasilkan tidak akan jauh berbeda atau sama dengan ketika aplikasi HTTP server menggunakan protokol TCP.

3.3 Batasan Perancangan

Batasan-batasan dalam melakukan perancangan sistem terbagi atas batasan yang bersifat teoritis dan batasan yang bersifat teknis.

1. Batasan Teoritis

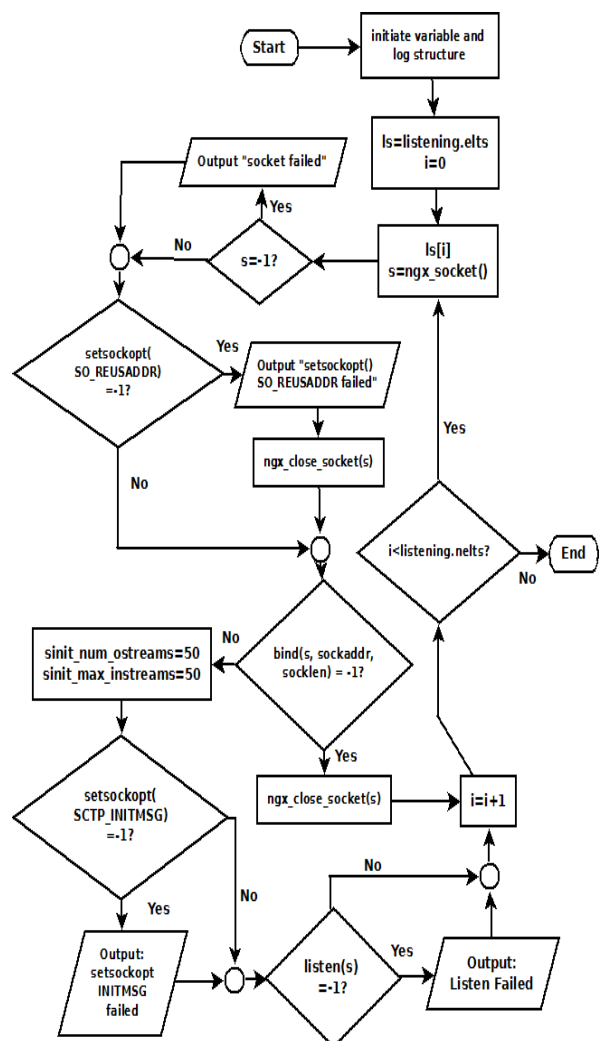
- Kemampuan protokol SCTP yang digunakan hanya kemampuan multistreaming seperti yang telah dijelaskan pada bab dasar teori.
- Inisiasi jumlah maksimal stream ditetapkan oleh server dan tidak melayani permintaan perubahan jumlah maksimal stream dari client. Dalam hal ini client hanya bisa menggunakan batasan maksimal jumlah stream yang telah disediakan oleh server.
- Hasil akhir dari perancangan ini adalah aplikasi HTTP server yang hanya mampu menggunakan protokol SCTP sebagai protokol transportnya.

2. Batasan Teknis

HTTP server Engine X yang digunakan adalah versi 0.8.31 dan HTTP client yang digunakan adalah phhttpget versi 0.2 yang telah mendukung protokol HTTP 1.1. Sedangkan modul kernel untuk protokol SCTP menggunakan LKSCTP versi 1.0.8.

3.4 Perancangan

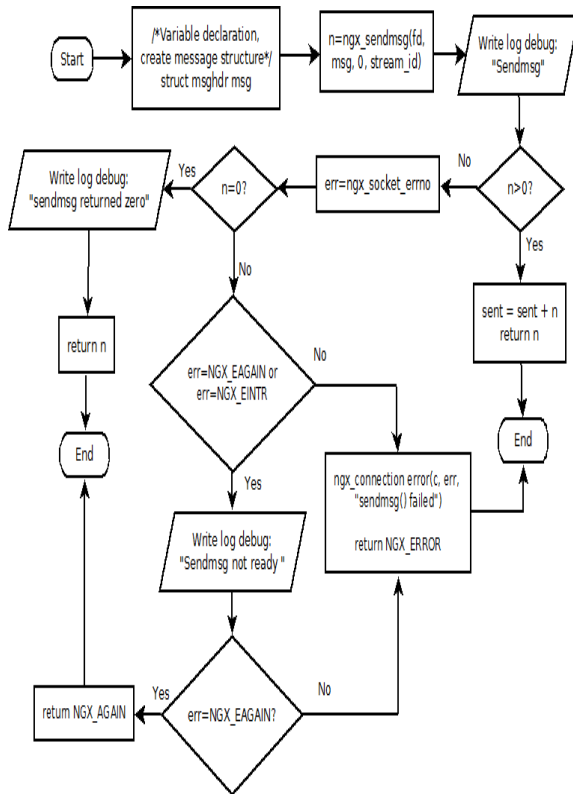
1. Perancangan fase pembentukan Asosiasi



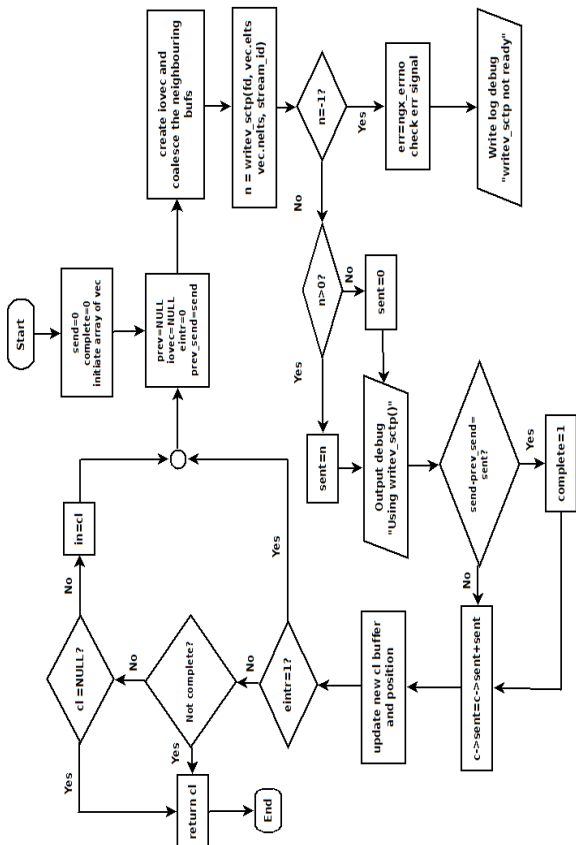
Gambar 1: Flowchart fungsi ngx_open_listening_sockets

2. Perancangan fase transmisi data

a. Pengiriman response HTTP

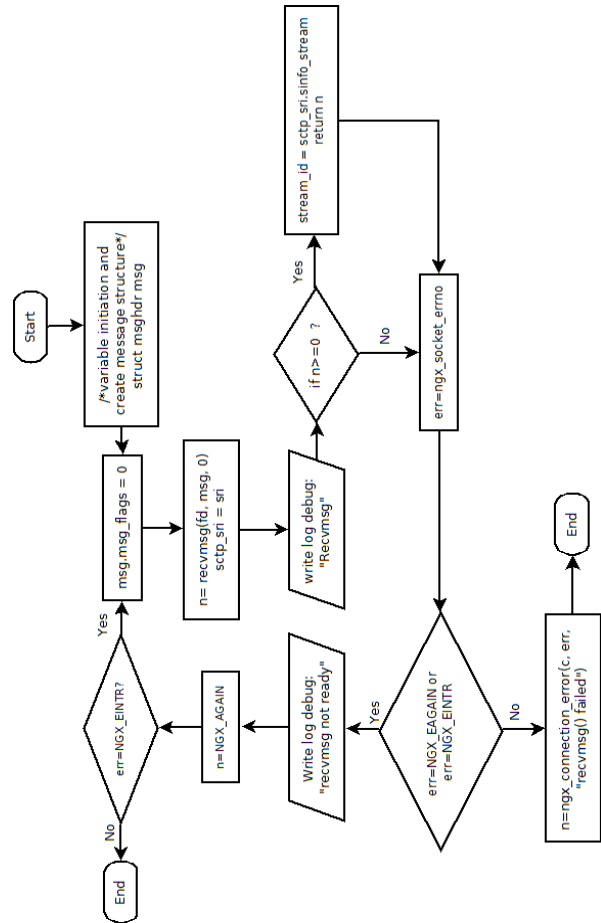


Gambar 2: Flowchart fungsi ngx_unix_send_sctp



Gambar 3: Flowchart fungsi ngx_writev_chain_sctp

b. Membaca request HTTP



Gambar 4: Flowchart ngx_unix_recv_sctp

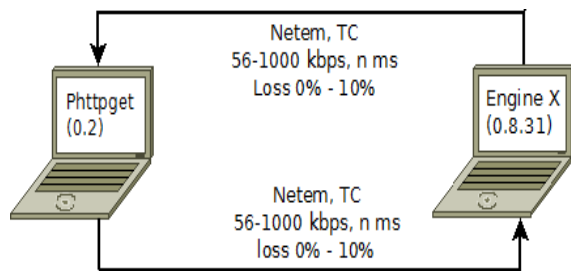
4. Pengujian dan Analisa

4.1 Skenario Pengujian

Pengujian performansi ini dilakukan dengan mengikuti beberapa tahapan sebagai berikut:

1. Perancangan Jaringan

Perancangan jaringan dilakukan dengan menghubungkan pc server dengan pc client menggunakan sebuah kabel UTP. Pada sisi server dilakukan instalasi dua versi aplikasi HTTP server Engine X (nginx). Aplikasi nginx yang menggunakan protokol SCTP akan menunggu asosiasi pada port 81 sedangkan aplikasi nginx yang menggunakan protokol TCP akan menunggu koneksi dari client pada port 80. Pada PC client dilakukan instalasi dua aplikasi phttpget. Masing-masing aplikasi phttpget hanya mendukung salah satu protokol transport saja, TCP atau SCTP. Simulasi bandwidth yang menunjukkan kondisi jaringan yang berbeda dilakukan dengan menggunakan aplikasi TC dan netem.



Gambar 5: Skenario jaringan untuk pengujian

Pada gambar 5 diatas menunjukkan bentuk jaringan yang digunakan dalam pengujian. Ukuran bandwidth yang digunakan bervariasi dari 56 kbps, 384kbps, dan 1Mbps. Ukuran bandwidth tersebut secara berurutan menunjukkan pengguna yang menggunakan koneksi modem dial-up PSTN, modem adsl, dan HSDPA. Pengujian ini adalah untuk melakukan simulasi pengguna yang mengakses situs *en.wikipedia.org* pada lokasi yang sama dengan menggunakan koneksi internet yang berbeda. Untuk masing-masing ukuran bandwidth yang digunakan dilakukan juga simulasi delay jaringan yang terjadi ketika lokasi pengguna berada di Bandung dan melakukan akses ke situs tersebut.

2. Pengukuran Page Response Time

Waktu pengukuran page response time dihitung mulai dari pengiriman request pertama yang dilakukan oleh client ke server sampai dengan response dari request terakhir telah diterima sepenuhnya oleh client. Pesan request pertama yang dikirimkan ke server adalah pesan untuk mengambil halaman web dan kemudian request selanjutnya dilakukan secara pipeline (melalui protokol TCP) atau multistreaming (melalui protokol SCTP) yang bertujuan untuk mengambil objek-objek web yang ada di dalam halaman tersebut. Kemudian dilakukan pencatatan waktu dari hasil pengujian tersebut. Asumsi yang dibuat adalah pada waktu melakukan penguraian isi halaman web tidak dilakukan secara otomatis dikarenakan web client phhttpget tidak bisa melakukan hal tersebut. Untuk melakukan request HTTP maka dibuat urutan URI dari objek-objek web tersebut yang akan diunduh oleh aplikasi phhttpget. Urutan yang pertama adalah URI dari halaman web dan selanjutnya secara berurutan merupakan URI dari masing-masing objek web yang ada di dalam halaman web tersebut. Data-data pembentuk halaman web yang digunakan merupakan data rekaan. Ukuran dari data-data web tersebut berkisar antara 100byte-20Kbyte.

3. Pengukuran Throughput

Throughput merupakan bandwidth aktual yang terukur pada suatu ukuran waktu tertentu. Pengukuran throughput dapat dirumuskan sebagai berikut:

$$\text{Throughput (kbit/detik)} = \frac{\sum n}{t \times 1000} \times 8 \text{ bit} \quad (1)$$

n: adalah total ukuran dari objek-objek web yang diunduh (dalam satuan byte).

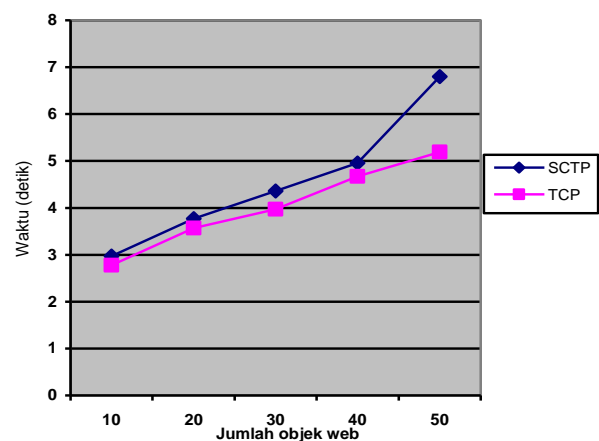
t: waktu yang dibutuhkan untuk mengunduh seluruh objek-objek web (dalam detik).

Pengukuran throughput dilakukan dengan menggunakan skenario jaringan dan jumlah variasi ukuran objek-objek web sama seperti skenario yang dilakukan untuk mengukur page response time.

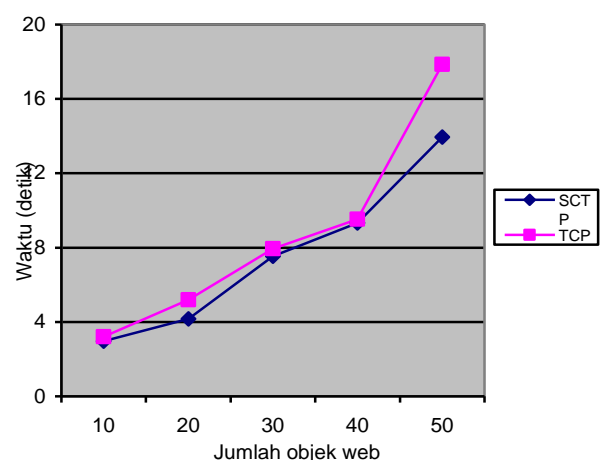
4.2 Analisa Pengujian Performansi

1. Analisa pengujian page response

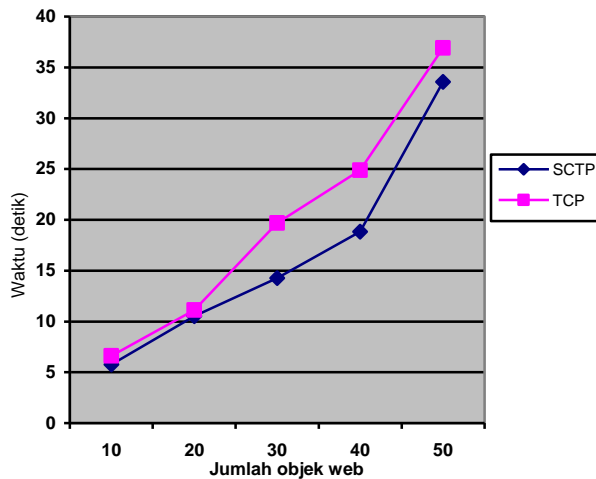
Berikut ini adalah diagram perbandingan page response time untuk ukuran bandwidth 1Mbps dengan loss 0%, 3%, 10% yang dihasilkan oleh web server Engine X yang menggunakan protokol TCP dan protokol SCTP:



Gambar 6: Diagram perbandingan waktu untuk bandwidth 1Mbps dengan loss 0%



Gambar 7: Diagram perbandingan waktu untuk bandwidth 1Mbps dengan loss 3%

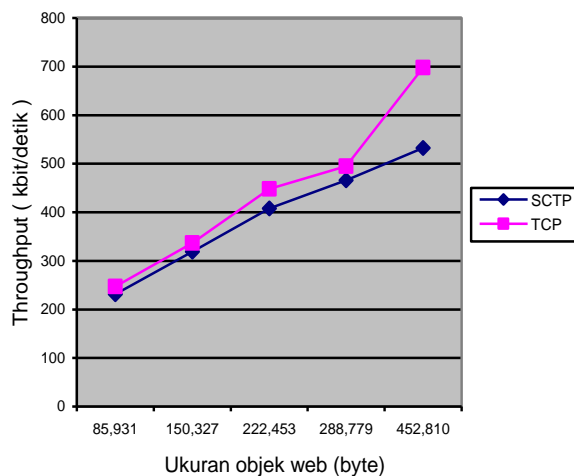


Gambar 8: Diagram perbandingan waktu untuk bandwidth 1Mbps dengan loss 10%

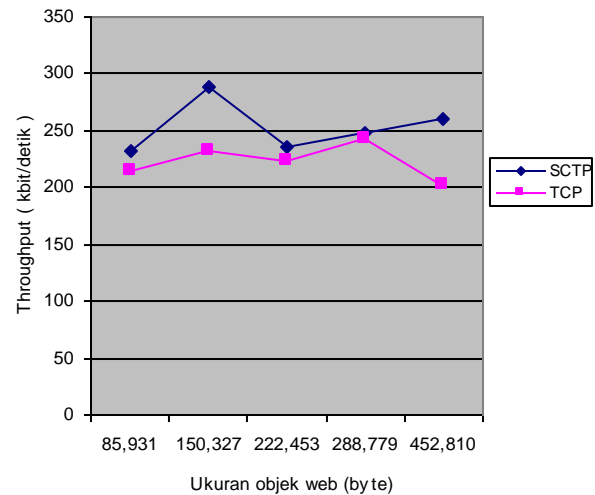
Berdasarkan dari diagram grafik diatas dapat dilihat bahwa kecenderungan waktu yang diperlukan oleh web client dan web server untuk melakukan transaksi HTTP secara persistent dan pipeline dengan menggunakan protokol TCP lebih cepat jika dibandingkan dengan menggunakan protokol Sctp yang dilakukan secara persistent dan multistreaming untuk setiap kondisi jaringan yang baik, dalam hal ini jaringan tanpa adanya packet loss. Namun ketika kondisi jaringan terdapat packet loss, maka waktu yang diperlukan oleh web client dan web server untuk melakukan transaksi HTTP akan cenderung lebih lama.

2. Analisa pengujian throughput

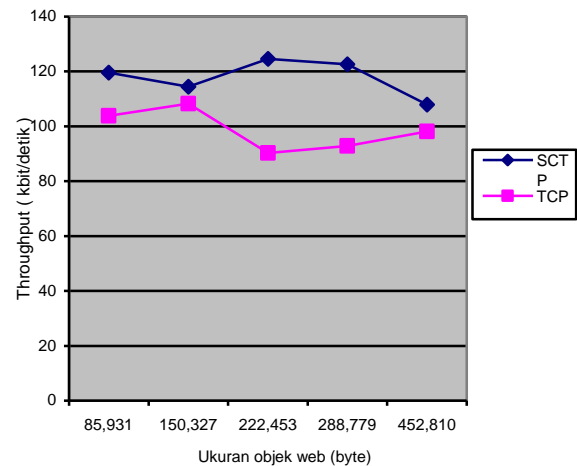
Berikut ini adalah diagram garis hasil pengukuran throughput untuk ukuran bandwidth 1 Mbps dan variasi loss 0%, 3%, dan 10%.



Gambar 9: Perbandingan throughput untuk bandwidth 1Mbps dengan loss 0%



Gambar 10: Perbandingan throughput untuk bandwidth 1Mbps dengan loss 3%

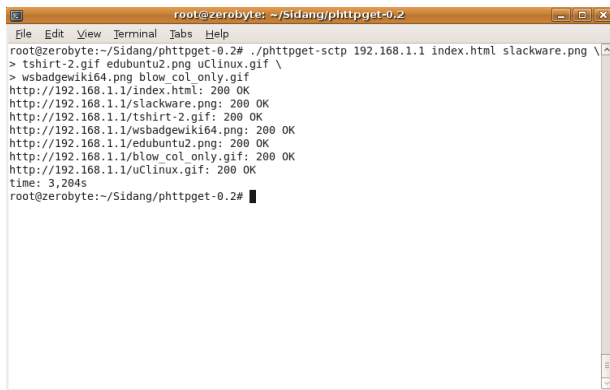


Gambar 11: Perbandingan throughput untuk bandwidth 1Mbps dengan loss 10%

Dari diagram garis-digaris diatas dapat disimpulkan bahwa pada kondisi jaringan tanpa packet loss, throughput yang dihasilkan oleh protokol TCP cenderung lebih besar dibandingkan protokol Sctp. Namun untuk kondisi jaringan yang terdapat packet loss maka throughput yang dihasilkan protokol TCP akan cenderung lebih kecil dibandingkan protokol Sctp.

3. Analisa pengujian HOL blocking

Selain dari dua pengujian performansi sebelumnya dapat disimpulkan juga bahwa penggunaan kemampuan multistreaming dari protokol Sctp dapat memberikan keuntungan untuk mengurangi terjadinya kondisi HOL blocking seperti yang ditunjukkan pada gambar berikut ini:



```
root@zerobyte: ~/Sidang/phttpget-0.2
File Edit View Terminal Tabs Help
root@zerobyte:~/Sidang/phttpget-0.2# ./phttpget-sctp 192.168.1.1 index.html slackware.png
> tshirt-2.gif edubuntu2.png uclinux.gif \
http://192.168.1.1/index.html: 200 OK
http://192.168.1.1/slackware.png: 200 OK
http://192.168.1.1/tshirt-2.gif: 200 OK
http://192.168.1.1/wsbadgiewiki64.png: 200 OK
http://192.168.1.1/edubuntu2.png: 200 OK
http://192.168.1.1/blow_col_only.gif: 200 OK
http://192.168.1.1/uclinux.gif: 200 OK
time: 3.2045
root@zerobyte:~/Sidang/phttpget-0.2#
```

Gambar 12: Transaksi HTTP melalui protokol SCTP antara phttpget dan Engine X

Pada gambar diatas menunjukkan transaksi HTTP yang dilakukan melalui protokol SCTP pada simulasi jaringan yang dikondisikan terdapat paket data yang hilang. Pada gambar tersebut web client phttpget melakukan urutan request yang dilakukan secara multistream kemudian response dari masing-masing request HTTP tersebut ditampilkan ke layar jika semua bagian dari response tersebut telah sepenuhnya diterima. Dari gambar tersebut juga ditunjukkan bahwa response-response HTTP yang ditampilkan tidak dalam kondisi terurut sesuai dengan urutan request yang dikirimkan. Berbeda dengan ketika menggunakan protokol TCP, urutan response yang diterima aplikasi client akan selalu sama dengan urutan request yang dikirimkan.

5. Kesimpulan dan Saran

1. Kesimpulan

- Penggunaan protokol SCTP dapat mengurangi terjadinya kondisi HOL blocking pada transaksi HTTP terutama dalam proses pengiriman pesan-pesan response dari server.
- Web client yang mengirimkan request-request HTTP secara multistreaming dengan menggunakan protokol SCTP, dalam hal ini masing-masing request dikirimkan ke dalam stream-stream yang berbeda, akan menerima response-response HTTP dalam keadaan terurut atau tidak terurut. Sedangkan request-request HTTP yang dilakukan secara pipeline oleh web client dengan menggunakan protokol TCP akan selalu menerima response-response HTTP dari server dalam keadaan terurut sesuai dengan urutan dari request yang dikirimkan.
- Penggunaan stack protokol SCTP (LKSTCP) pada kernel linux jika dibandingkan dengan penggunaan stack protokol TCP (New Reno) akan menghasilkan performansi yang sedikit berbeda ketika dilakukan pengujian yang dilihat dari sisi page response time. Waktu yang dibutuhkan protokol SCTP untuk melakukan transaksi secara persistent dan multistreaming menghasilkan waktu yang lebih lama

dibandingkan jika menggunakan protokol TCP yang dilakukan secara persistent dan pipeline untuk kondisi jaringan tanpa adanya paket data yang hilang. Namun dalam kondisi jaringan yang terdapat paket data yang hilang, maka transaksi HTTP melalui protokol SCTP akan menghasilkan waktu yang lebih cepat dibandingkan transaksi HTTP melalui TCP.

- Dalam kondisi jaringan tanpa adanya paket data yang hilang maka throughput yang dihasil dari transaksi HTTP melalui protokol TCP akan lebih besar, Namun untuk kondisi jaringan yang terdapat paket data yang hilang maka throughput yang dihasilkan oleh protokol SCTP akan lebih besar dibandingkan TCP.

2. Saran

- Pengaturan jumlah batasan maksimal stream yang akan digunakan bisa dijadikan sebagai variabel yang dapat diatur di dalam berkas konfigurasi dari aplikasi web server Engine X. Sehingga pengguna tidak perlu melakukan kompilasi ulang kode sumber aplikasi setiap akan mengubah jumlah maksimal stream yang dibutuhkan.
- Dalam tugas akhir ini dilakukan modifikasi pada beberapa fungsi dari kode sumber aplikasi Engine X sehingga aplikasi tersebut hanya bisa mendukung protokol SCTP sebagai protokol transportnya. Tugas akhir ini dapat dikembangkan lagi sehingga aplikasi web server dapat mendukung penggunaan protokol SCTP dan TCP sebagai protokol level transportnya.

Daftar Pustaka:

- [1] Fielding, R. (et. al.), 1999, *Hypertext Transfer Protocol*, RFC 2616 [Online], Tersedia: <http://tools.ietf.org/html/rfc2616> [22 Juli 2009]
- [2] Hemminger, Stephen, 2005, Network Emulation with NetEm, Dalam Linux Conf Au [Online], Tersedia: http://developer.osdl.org/shemminger/LCA2005_paper.pdf
- [3] King, Andrew B, 2008, Average Web Page Triples Since 2003, [Online], Tersedia: <http://www.websiteoptimization.com/speed/tweak/average-web-page/> [26 Mei 2010]
- [4] Knutson, Karl dan La Monte H. P. Yarroll, 2001, *Linux Kernel SCTP: The Third Transport*, Dalam Ottawa Linux Symposium 2001 [Online], Tersedia: <http://old.lwn.net/2001/features/OLS/pdf/pdf/sctp.pdf> [22 Juli 2009]
- [5] Natarajan, Preethi (et. al.), 2006, *SCTP: An innovative transport layer protocol for the web*, Dalam 15th International Conference on World Wide Web [online], Tersedia: <http://www.cis.udel.edu/~nataraja/papers/www2006.pdf> [23 Juli 2009]

- [6] Natarajan, P. (et. al.), 2009, *Using SCTP as a Transport Layer Protocol for HTTP*, [Online], Tersedia: <http://tools.ietf.org/html/draft-natarajan-http-over-sctp-02> [12 Juli 2009]
- [7] Peterson, Larry L. dan Bruce S. Davie, 2000, *Computer Network: A Systems Approach. (2nd ed.)*, San Fransisco: Morgan Kaufmann Publishers.
- [8] Shiflett, Chris, 2003, *HTTP Developer's Handbook*, Indianapolis: Sams publishing.
- [9] Stevens, W. Richard (et. al.), 2003, *Unix Network Programming: The Sockets Networking*. 1(3rd ed.), Boston: Addison Wesley.
- [10] Stewart, R, 2007, *Stream Control Transmission Protocol*, RFC 4960 [Online], Tersedia: <http://tools.ietf.org/html/rfc4960> [13 Juli 2009]
- [11] Stewart, Randall (et. al.), 2008, *Multistreamed Web Transport for Developing Regions*, Dalam ACM SIGCOMM Workshop on Networked Systems for Developing Regions (NSDR) 2008 [online], Tersedia: <http://www.cis.udel.edu/~nataraja/papers/nsdr2008.pdf> [23 Juli 2009]
- [12] Stewart, R. (et. al.), 2009, *Socket API Extension for Stream Control Transmission Protocol (SCTP)*, [Online], Tersedia: <http://tools.ietf.org/html/draft-ietf-tsvwg-sctpsocket-19> [22 Juli 2009]
- [13] Wall, Kurt (et. al.), 1999, *Linux Programming Unleashed*, Indianapolis: Sams Publishing.